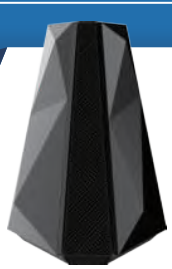
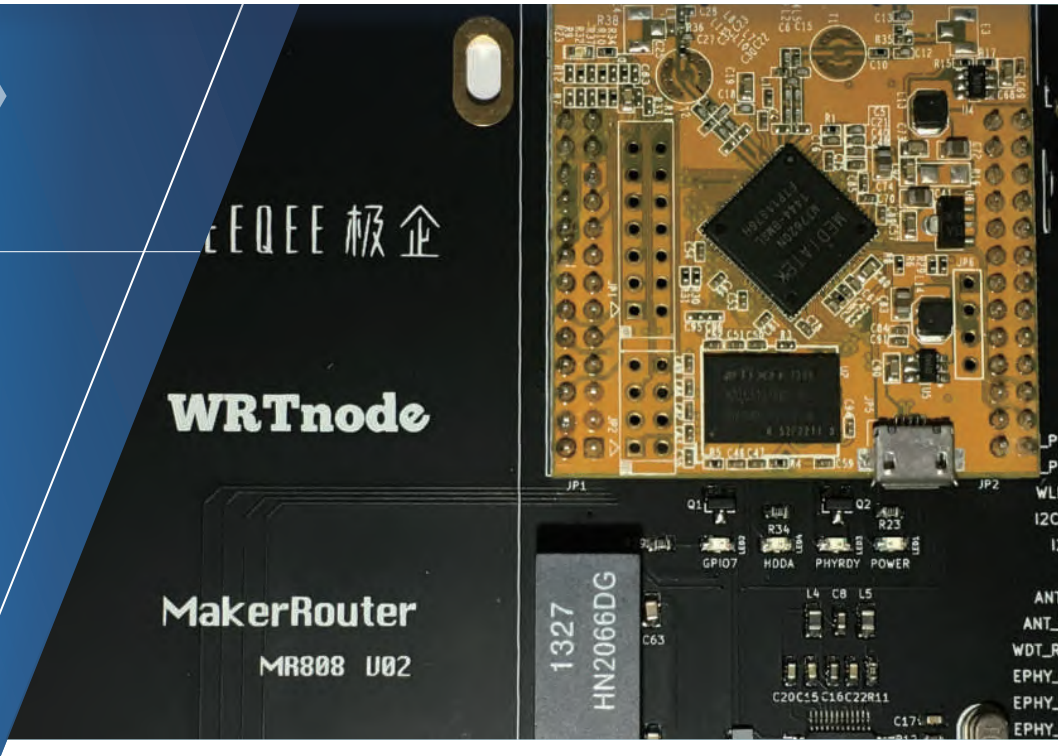


I'M MAKER 创客

OpenWrt 智能路由系统开发

跟 hoowa 学智能路由

王伟 孙冰 刘龙 著



中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

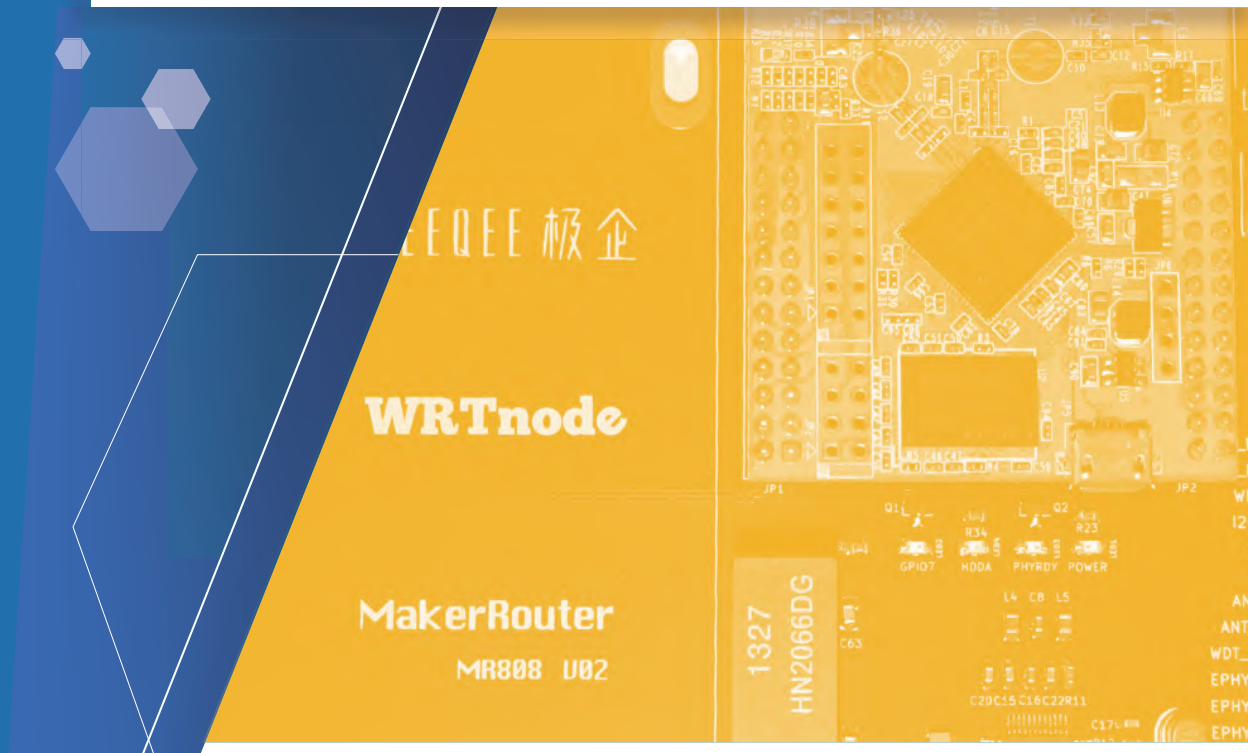
I'M MAKER 创客

OpenWrt

智能路由系统开发

跟 hoowa 学智能路由

王伟 孙冰 刘龙 著



人民邮电出版社
北京

图书在版编目（CIP）数据

OpenWrt智能路由系统开发 : 跟hoowa学智能路由 / 王伟, 孙冰, 刘龙著. — 北京 : 人民邮电出版社, 2018.3
（i创客）
ISBN 978-7-115-47711-8

Ⅰ. ①O… Ⅱ. ①王… ②孙… ③刘… Ⅲ. ①Linux操作系统－应用－路由器－系统开发 Ⅳ. ①TP316.85 ②TN915.05

中国版本图书馆CIP数据核字(2018)第002679号

内 容 提 要

本书汇集了各类基于OpenWrt系统的路由器、物联网、智能家居方向的技能和行业内知名企业的技术经验。本书区别于传统嵌入式系统书籍，并没有一上来就介绍那些晦涩的软硬件知识，本书所介绍的内容门槛更低，更适合读者循序渐进地完成学习。书中使用更轻松的表达方式，让读者开开心心就可以完成智能路由器的开发。

由于路由器大量采用了开源操作系统，哥伦比亚大学法学院教授Eben Moglen向路由器厂商提出了开源路由器系统部分的要求，就这样，2004年OpenWrt诞生了White Russian版，之后每一两年便发布一个最新的版本。今天，OpenWrt系统在智能路由器行业已是事实性标准，大量的芯片厂商及路由器厂商（包括小米、极路由、极企科技等知名互联网科技企业）均采用该系统作为路由器或智能家居控制中心，即使宣称没有使用OpenWrt的厂商，也多是在该系统基础之上进行了业务定制。

OpenWrt作为开源组织采取松散管理，很难找到一份比较系统的专业技术文档用于爱好者学习。因此2012年由hoowa发起、众多资深技术“大神”参与的“跟hoowa学智能路由”在线培训开始启航，本书所有内容均原滋原味来源于此。本书遵循“学中做，做中学”的理念，将内容分为19章，读者可以跟着本书内容，边学边做，最终完成一个智能路由器的开发。

本书主要从OpenWrt这个系统的角度来介绍，但书中内容绝不局限于路由器本身，通过阅读本书，读者可以掌握智能家居、智能玩具、工业控制等各类不同设备的玩法。本书适合计算机或电子类相关学生、从事IT行业的人员、智能设备或OpenWrt系统的发烧友、各智能设备厂商员工阅读。

◆ 著　　王　伟　孙　冰　刘　龙
责任编辑　周　明
责任印制　周昇亮

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京画中画印刷有限公司印刷

◆ 开本：700×1000　1/16
印张：15.25　　2018 年 3 月第 1 版
字数：332 千字　　2018 年 3 月北京第 1 次印刷

定价：79.00 元

读者服务热线：(010)81055339　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

关于本书

本书原名为《跟 hoowa 学做智能路由》，产生自 2014 年的春天，断断续续写了 3 年之久，在这过程中，有大量的好友参与。在爱好者的支持下，《跟 hoowa 学做智能路由》一共更新了 20 章，还建立了第一期在线学习版，吸引了大量的爱好者参加学习。我们在《跟 hoowa 学做智能路由》教程基础上对其进行了修改和完善，并重新审视了智能硬件、智能路由、智能家居、物联网之间的关联，协同“极企科技”等公司，共同推出了现在这本《OpenWrt 智能路由系统开发》。

智能路由这个行业的人才很少，希望本书可以帮助各种嵌入式开发工程师、热爱智能路由开发或想在这方面有所成就的读者。

读者对象

本书区别于传统嵌入式类书籍，并没有一上来就介绍那些晦涩的软硬件知识，这本书所介绍的内容门槛更低，更适合读者循序渐进地完成学习。书中使用更轻松的表达方式，让读者开开心心地完成智能路由的开发。

学做智能设备的几大误区：

（1）需要具备硬件知识：学习智能设备上的软件开发和硬件调试确实需要具备一定的硬件知识，但是所需具备的硬件知识非常非常少，少到几乎可以在几小时之内掌握，因此阅读本书的读者不需要事先具备硬件知识，但是我们希望读者具备良好的动手能力。

（2）需要懂嵌入式开发，能写驱动程序：完全不需要，本书中 Linux 相关的知识跟我们计算机上的大部分知识都是通用的，考虑到有很多读者连 Linux 都没有使用过，书中也加上了 Linux 部分的内容让读者学习。

（3）需要具备多年的开发经验：也是不需要的，本书推荐读者熟悉某种编程语言，这样可以方便理解各类知识，但是对编程语言以及平台没有任何要求，随便哪一种都可以，即使是 GWBASIC 这种语言都行。

推荐学习者类型：

本书主要从 OpenWrt 这个系统的角度进行介绍，但书中内容绝不局限于路由器本身，通过阅读、学习本书内容，读者可以掌握智能家居、智能玩具、工业控制等各类不同设备的玩法。

本书读者对象如下：

- 计算机或电子类相关学生
- 从事 IT 行业的人员
- 智能设备或 OpenWrt 系统的发烧友
- 各智能设备厂商员工

本书内容划分

本书遵循“学中做，做中学”的理念，将内容分为 19 章，读者可以跟着本书内容，边学边做，最终完成一个智能路由器的开发。

第 1 章　路由器概述：介绍了传统路由器与智能路由器以及它们的区别。

第 2 章　OpenWrt 介绍：介绍了什么是 OpenWrt、哪些产品用了 OpenWrt、OpenWrt 是如何支持路由器 CPU 的。

第 3 章　智能路由器硬件：介绍了 MIPS 处理器架构、国内常见路由的配置、常见路由器芯片厂商、MT7620 芯片、MR808MR808 开发板和 AXWRT 系统。

第 4 章　智能路由开发环境搭建：介绍了智能路由开发环境的搭建，包括虚拟机和操作系统的安装以及 SSH 远程登录。

第 5 章　基本知识：介绍了常用 Linux 命令、Vi 编辑器的使用和基础 Shell 编程。

第 6 章　OpenWrt 刷机：介绍了几种不同 OpenWrt 刷机方式，第一种是通过串口进入系统，第二种是 U-Boot 的 TFTP 刷机，第三种是 U-Boot 的 Web 刷机。

第 7 章　分区、外网与 OPKG：介绍了路由器上的分区、查看系统 MTD 的分区、文件系统、挂载情况查看分区、透明挂载 / 根目录、基本系统演示和 OPKG 软件包管理。

第 8 章　UCI 命令系统：介绍了 UCI 命令、UCI 文件、UCI 流程和文件语法以及 UCI 综合实例。

第 9 章　智能路由配置：介绍了网络配置文件、配置 WAN 口外网、配置 LAN 口服务、DHCP 服务以及如何连接外网。

第 10 章　防火墙与 UPnP：介绍了防火墙命令、防火墙配置和实现 UPnP。

第 11 章　时区、服务、SSH 与 SCP：介绍了信息和时区、服务管理、SSH 远程登录、SCP 文件复制。

第 12 章　存储器扩展：介绍了存储器扩展实现原理、存储器选型、rootfs_data 数据迁移和挂载的注意事项。

第 13 章　网络存储与文件共享：介绍了 Windows 文件共享实现、FTP 文件共享实现。

第 14 章　智能路由器媒体中心：介绍了 USB 声卡驱动、安卓手机远程播放音乐、苹果 AirPlay 播放音乐。

第 15 章　智能路由器编程工具：介绍了 Web+PHP 建站系统、SQLite3 与 MySQL 数据库应用。

第 16 章　智能路由器服务软件：介绍了 VPN(PPTP) 服务器与客户端、USB 摄像头、迅雷远程下载、WiFiDog 广告路由。

第 17 章　编译定制 OpenWrt 固件：介绍了 OpenWrt 编译环境、OpenWrt 固件生成、OPKG 自定义软件仓库。

第 18 章　编译定制 IPK 软件包：介绍了 IPK 软件包与 OPKG 命令、IPK 软件包规范、mytest 简易软件包、软件包示例 xware、软件包示例 shairport 和 IPK 软件包杂说。

第 19 章　智能路由外设应用实例：介绍了 OpenWrt 对 LED、按钮和继电器的支持。

本书所有资源下载地址为：http://www.maker1000.com/makerouter/，此外，在各章节中还给出了具体网盘下载地址。

OpenWrt 社区对中国开发者寄语

I would like to say that, the OpenWrt project is very very happy about the work that takes place in China among the Chinese developers, and we are really like to see that they become more visibly access in the larger OpenWrt development, see more patches submitted, we would like see them talking more active part into discussion. And of course I know that there are some language barriers, but again that's something we can work with, so in that sense, I see a very positive relationship. And there is just one thing to do and to keep on doing the work and doing more of it in common.

我想说，很高兴看到 OpenWrt 项目在中国得到了开发者的广泛应用，希望看到你们更活跃地推动 OpenWrt 的发展，多向 OpenWrt 项目提交代码，踊跃参与在线讨论。尽管很多开发者可能存在语言障碍，但我很乐观。因为这都不是事儿，只要一起动手做，一定能有所斩获。

——OpenWrt 社区经理 Gregers Petersen

致谢

本书撰写离不开众多小伙伴的密切合作与大力支持，在此感谢以下公司（排名不分先后）：极企科技 www.geeqee.com（本书所使用的定制版操作系统 AXWRT 的著作权所有者）、硬创邦 www.ingchuang.com（在线推广和发行）、www.FlamingoEDA.com（首批 MR808 底板设计生产商）和 www.WRTnode.com（首批核心板生产商）。感谢几十人的工作团队、500 位爱好者作为最早的贡献者让本书得以顺利完成。还要感谢王明海先生对于本书 16.2 节 ~16.4 节贡献了大量内容，王明海先生是工业自动化和家庭智能路由器研发总监，从事嵌入式研发超过 10 年。

本书编写过程中还参考了大量嵌入式系统方面的书籍和资料，在此对相关作者表示感谢。书中难免存在遗漏和错误的地方，如读者发现有这方面的问题，请给予指出，必将采纳。我们的联系方式为 E-mail：hoowa.sun@gmail.com。

CONTENTS

目 录

第1章 路由器概述

1.1 传统路由器

全世界近 80% 的路由器是在中国生产的，其中约有 50% 的路由器是在中国研发的，传统的路由器厂商使用几大路由器芯片厂商提供的设计方案进行产品设计。这些芯片厂商会给路由器厂商提供资料、电路板设计原理图、软件 SDK。其实很多人可能不知道，芯片厂家所提供的资料完善到你难以想象的地步，只需要有一台老旧计算机，随便一编译，自己做个板子，再随意弄个外观就可以完成路由器设计。因此，传统路由器厂商能用来提升营收和竞争力的手段就只剩下价格和外观了，为此这个行业进入了一种越做越烂的循环模式。而在这之外，一些在成本上不具备优势的小厂商，为了获得利润和生存空间，普遍使用违规的功率放大装置，做出让你觉得信号很强的产品，成为了灰色地带。

由于国内的传统路由器厂商过于注意外观（比如路由器必须有天线，因为客户觉得有天线信号才会强），不注重软件（芯片厂家连 Web 都做了，没自己啥事），所以普遍不重视技术培养和研发，如今在智能路由大行其道的时候，他们往往陷入了被动。像 TP-Link、磊科这种大厂已经在加强这方面投入，取得了可喜的成绩，值得肯定。

1.2 智能路由器

所谓智能路由器，有人定义为它像个人计算机一样，具有独立的操作系统，用户可自行安装软件控制带宽、上网加速、过滤视频广告等，远不局限于无线上网的功能。它将一个功能单一的产品变成了一个平台，在这个平台上面，可以安装 App 插件来增加新的功能，例如游戏加速、下载加速和全球加速，同时还可以增加新的使用场景，比如用路由器无线播放音乐等。问题是，路由器这个多年未变的市场，为何会突然出现智能路由器的概念和产品？

智能路由器也就是智能化管理的路由器，通常具有独立的操作系统，可以由用户自行安装各种应用，自行控制带宽、在线人数、浏览网页和在线时间，同时拥有强大的 USB 共享功能，真正做到网络和设备的智能化管理。

本书将重心放在了智能路由上，智能路由从 2013 年开始逐渐升温，互联网公司开始关注传统路由器这个发展了十几年没有任何技术革新的产业，作为家庭、企业的入口设备和控制中心，路由器的可想象空间变得更大。由于路由器芯片的性能不断增强，现在我们所看到的路由器在性能和配置上已经接近十几年前的计算机，并且即将超越，同时具备了更低的功

率、更高的性能。

只要你有一款智能路由器，你就可以将它和任何一种智能设备连接在一起。当前大部分智能路由器采用了 OpenWrt 这个开源路由器操作系统作为自己的系统，然后对系统进行修改，做个有自主知识产权的产品。

1.3 传统路由器与智能路由器的区别

智能路由器，不同厂商的产品有不同的功能，有的有屏幕，有的可以实现网络加速，甚至有的可以直接播放音乐，但是如果要对传统路由器与智能路由器两类设备用一句话来区分，那就是“智能路由器可以单独安装软件”。

第2章 OpenWrt 介绍

2.1 什么是 OpenWrt

OpenWrt 和 DD-Wrt 等是现在很多智能路由设备的操作系统。虽然没有某一个厂商专门单独推出这样一个开源的产品，但 OpenWrt 在中国已经成为了智能路由器的标准配备。

来自美国的 OpenWrt 社区对 OpenWrt 做了一个简短而精准的说明：“OpenWrt 就是为嵌入式设备所研发的 Linux 发行版”。当年 Linksys 开放了一款路由器的源代码，然后就有不同的黑客对这个源代码打补丁用以实现不同的功能，最后就出现了针对不同市场的杂乱无章的路由器固件。

OpenWrt 选择了另外一条路，从开始的那一刻起，它就一点点把各种软件和玩意加到系统中，采用了一个非常灵活并且开放式的方法，这样的结果就令全世界所有的厂商和爱好者都能加强 OpenWrt 的功能，而开放式架构也令 OpenWrt 支持数量繁多的芯片，从 x86 到 ARM、MIPS 等各种稀奇古怪的芯片。而开发者使用 OpenWrt，只需要直接通过简单的编译，就可以将一套软件编译出不同芯片的版本，烧写到自己的路由器上就拥有了一台 Linux 服务器。OpenWrt 还支持一种称为 OPKG 的增强型安装技术，OPKG 是 OpenWrt Package Management 在 OpenWrt 下的一款轻量级软件包管理工具，OPKG 使用起来就像 CentOS 下的 YUM 一样，一个命令就可以将已预编译好的软件安装到系统中。

从 2005 年 2 月截至今日，OpenWrt 已经支持了一千多种软件，并且自己移植软件到 OpenWrt 中变得非常容易。OpenWrt 系统高度模块化，有人说我的产品是深度定制的 OpenWrt，现在跟随我们的步伐把本书学完，人人都可以定制 OpenWrt 系统，人人都可以开发智能路由器或智能家居产品。

2.2 哪些产品用了 OpenWrt

- 极路由：极壹 S
- 极路由：极硬货
- 极路由：极贰
- 极路由：极壹
- 小米：小米路由器
- 小米：小米路由器 mini

- 盛大果壳：魔豆路由（类 OpenWrt）
- 联想：新路由

以上只列出了市面上较为流行的智能路由器，实际使用的远超过此列表。而以上列表中的大部分硬件都与本书所讲的 MR808 开发板兼容。

2.3 OpenWrt 是如何支持路由器 CPU 的

一般，这种支持要么是水平极高并且是芯片厂合作伙伴的人实现的，要么是芯片厂的内部人士实现的。想要支持路由器 CPU 说起来并不难，只需要两部分的支持。一是在汇编层面让 OpenWrt 编译时支持所属路由器 CPU 的指令集，方便编译出软件，在路由器上运行。这部分基本上是可以支持的，现在大部分路由器采用的都是 MIPS 指令集，这是 Linux 已经支持的。二是驱动程序了，外围的厂商也不容易写出驱动程序，所以大部分驱动程序要么是芯片厂直接提供的，要么是有更亲密关系的厂商写的。

有一个好消息，目前市面上的路由器 CPU 厂商都已经开始重视 OpenWrt 系统，虽然它们还不愿意做到首先支持 OpenWrt，但在芯片上市半年左右都会推出对 OpenWrt 的支持，值得赞扬！

第3章 智能路由器硬件

3.1 MIPS 架构

现在的智能路由器，其核心部分与我们的计算机没有太大的区别，无非就是 CPU 配合存储器，加上一些外围电路，没什么特别“高大上”的地方。

近 10 年来，路由器所采用的 CPU 主要是基于 MIPS 架构设计的。说到 MIPS，可能很多人感到陌生；说到龙芯，了解的人就比较多了，龙芯便是 MIPS 架构下的一种处理器。

MIPS 架构是由斯坦福大学的教授约翰·轩尼诗在闲得无聊的时候带领他的团队开发出来的一款处理器。在随后的 3 年时间里，他创立了 MIPS 科技公司。MIPS 采用了 RISC 精简指令集设计技术，RISC 是精简指令集的简称，是芯片的一种设计模式，相比 CISC（复杂指令集）具有更高的执行效率。MIPS 架构处理器在当时刷新了 32 位处理器性能的历史。在 2002 年，中国研发龙芯处理器，龙芯处理器采用了 MIPS 架构但是没有经过 MIPS 公司授权而遭到起诉，2009 年与 MIPS 公司和解。

MIPS 架构在历史上曾经是三国时期（x86、MIPS、PowerPC）的佼佼者，在 2000 年的时候，hoowa 记得国内可以买到一款叫作 Cobalt 的基于 MIPS 架构的服务器（Sun 公司为抢占市场收购了 Cobalt Network 公司）。

3.2 国内常见路由器的配置

国内常见路由器的配置见表 3-1。

表 3-1 国内常见路由器硬件配置

功能	配置
中央处理器架构	MIPSEL（目前市面上常见的路由器所采用）、ARM（较少见）
处理器主频	300 ~ 800MHz
内存类型	SDRAM / DDR1 / DDR2 / DDR3
闪存类型	Nor Flash / SPI Flash / Nand Flash
闪存容量	8 ~ 256MB
网络交换机	芯片内置或外挂

续表

功能	配置
Wi-Fi	芯片内置或外挂
操作系统	Linux SDK、VxWorks、eCos、OpenWrt

Tips

MIPSEL：MIPS 指令集的一种子集，采用了小端字节序，与 Intel x86 采用的一致。

ARM：智能手机常见处理器，由于集成度和成本控制不如 MIPS 架构，在路由器上使用得并不普遍。

Nor Flash：最早用于嵌入式的闪存技术，读取速度接近于内存，写入速度稍慢。

Nand Flash：目前使用最多的闪存技术，容量较大，在手机、U 盘中用得较多，在路由器中使用得较少。

SPI Flash：串行总线闪存技术，目前路由器使用得最多，读取速度比 Nor Flash 慢，但体积更小，成本更低。

CPU：路由器的主芯片，大部分市面上能见到的 CPU 使用的都是 BGA 封装或低成本的 QFN 封装。前者支持更多的接口，焊点在芯片下面，如果要手工拆换，需要使用热风枪（要求具有动手经验）。而 QFN 封装太脆弱了，很难更换。

内存：SDRAM 内存是早年路由器芯片所采用的内存，国际大厂已经很少生产，多是由一些小厂制作，速率较慢，容量不高。DDR/DDR2 内存是现在路由器所采用的主流内存。DDR3 内存对工艺要求更高，所以为了降低成本，路由器上使用得并不多。

观察各种芯片：路由器上有很多芯片，但如果对其分类，无非是 CPU、晶体振荡器、内存、闪存、供电、外围 I/O 接口电路，总体结构上来说趋于简单。甚至有厂商已经推出了内存都集成在 CPU 中的芯片。

3.3 常见路由器芯片厂商

目前路由器芯片（指主芯片 / 中央处理器 /CPU）普遍采用 SoC 设计，很多路由器芯片厂家喜欢把它叫作 RoC，这是个看起来很酷的名字。

这些厂家将内存控制、外围 I/O 接口、交换机芯片、Wi-Fi 芯片、加密芯片、SATA 接口、PCIE 接口、USB 接口等，你所能想到的所有硬件统统地集成到了一个手指甲大小的芯片当中，而其功耗却只有 1~3W，性能达到了 300~800MHz。同时只有少数厂家开始基于 ARM 开发路由器芯片，比如 Broadcom 公司的 BCM470X 系列芯片，采用的就是 ARM 架构，小米路由器第一代售价为 699 元的产品便是基于此芯片设计。

Atheros：全球路由芯片的顶级公司，其创始人是 MIPS 架构的发明者约翰·轩尼诗。该公司也是 Wi-Fi 标准的制定者之一，当前该公司芯片主要用于高端及企业级产品线，目

前 Atheros 已被高通收购。

Broadcom：美国博通公司，其无线路由芯片产品以稳定可靠著称。小米第一代路由器便是采用其 BCM470X 系列处理器，该处理器是 ARM 架构的，不过 Broadcom 其他的路由芯片大多是 MIPS 架构的。

Ralink：雷凌科技也是一家常年耕耘在路由器领域的芯片厂商，而且是目前国内出货量较大的家庭路由器芯片厂商，其芯片价格也是相对经济的。因此目前国内上市的家庭智能路由器产品大多采用了该公司方案。该公司目前已被联发科收购。

Tips

SoC：片载系统，表示在 CPU 内部集成了大量的其他芯片和功能，减少电路板外围芯片数量。

RoC：片载路由器，表示一个 CPU 就可以满足路由器所需要的最核心部分的硬件。

3.4 本书智能路由硬件

3.4.1 本书硬件的由来

“蛾儿雪柳黄金缕，笑语盈盈暗香去。众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。”

在 OpenWrt 官网上有一个兼容硬件列表，其中列出了大量的 OpenWrt 已兼容硬件。这些硬件需要两个操作，一个需要手工修改硬件，另一个需要购买，可是很多型号在中国是买不到的（其实很多在其他国家也买不到）。

在淘宝网上也有大量已经修改好了的 OpenWrt 路由器硬件在销售，这些硬件很多都是用旧货修改的，并且卖家所使用的系统都是由一些国内爱好者私人修改的，并没有开源，也不利于学习。

因此，经过本书的几个合作方一致同意，我们专门为 OpenWrt 设计一款硬件，让更多的人能够轻松入门。

Tips

OpenWrt 官方兼容硬件列表地址：http://wiki.openwrt.org/toh/start

3.4.2 MT7620 芯片

MT7620 芯片是联发科公司于 2012 年收购 Ralink 后推出的首款路由器主芯片（CPU），主频高达 600MHz，价格便宜量又足，虽然刚推出之后出现过驱动问题，经过两年的发展，已经成为了最成熟的路由器 CPU 选择之一，是国内智能路由厂商选择最多的芯片，也是我们今天的 MR808 开发板所使用的芯片。MT7620 系列分两个型号，分别为 MT7620N 与

MT7620A，这两款芯片的基本规格如表 3-2 所示。

表 3-2　MT7620N 与 MT7620A 基本规格比较

功能	MT7620N	MT7620A
CPU 指令集版本	MIPS24KEc	MIPS24KEc
主频	580MHz	580MHz
CPU 缓存	64KB+32KB	64KB+32KB
二级缓存	无	无
硬件转发	支持	支持 2Gbit/s
内存类型	SDRAM，DDR1，DDR2	DDR2
内存最大容量	64MB	256MB
NAND 闪存	不支持	最大支持 1GB
SD 卡接口	不支持	SD-XC（Class 10）
无线射频	2T2R 802.11n 2.4GHz	2T2R 802.11n 2.4GHz
PCI-E 接口	不支持	1x
USB2.0	1	1
交换机	5 口	5 口 +1 千兆 / 4 口 +2 千兆
I^2S、PCM	无	1
I^2C	1	1
GPIO	72	72

Tips

2T2R：无线的一种规格，两路接收，两路发射，实现了 300Mbit/s 的无线吞吐能力。

I^2S：一般用于 Hi-Fi 音箱的数字音频接口。

PCM：用于多路语音传输的音频接口。

3.4.3　MR808 开发板简述

MR808 开发板由 3 部分组成：核心板、底板和外围板。本书中全部知识都基于 MR808 开发板做过完整测试，但是并不代表学习 OpenWrt 必须购买 MR808 开发板，实际上这些知识都是 OpenWrt 系统的通用知识，如果要购买 MR808 可以访问 http://www.maker1000.com/makerouter/ 获得更多的信息。

核心板：采用 MT7620N 处理器，64MB 内存，16MB 闪存，具备大量开放式 PIN 接口，板载 300Mbit/s 无线网卡。核心板实物如图 3-1 所示。

底板：采用沉金工艺制作，具备 2 个 RJ-45 接口、两个 USB 接口、一个电源开关、一个电源接口。另外板载了 TTL 串口、3 个 ETHPHY 插针、1 个 USB 插针、14 个 GPIO 接口，用于扩展各类外围设备连接。底板实物如图 3-2 所示。

按钮板：轻触式按钮，配有专用连接线，线序定义是：黄色为 GPIO，红色为 3.3V，

黑色为 GND。按钮板实物如图 3-3 所示。

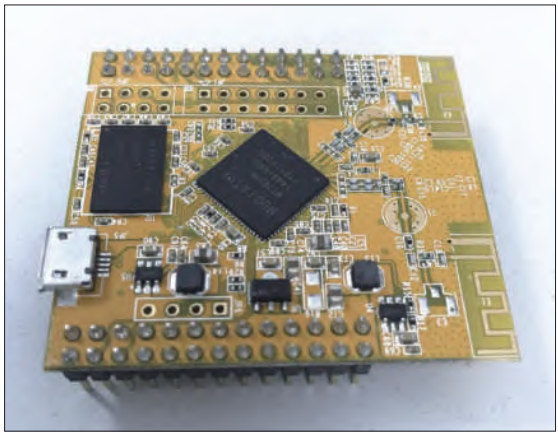


图 3-1 核心板实物图

图 3-2 底板实物图

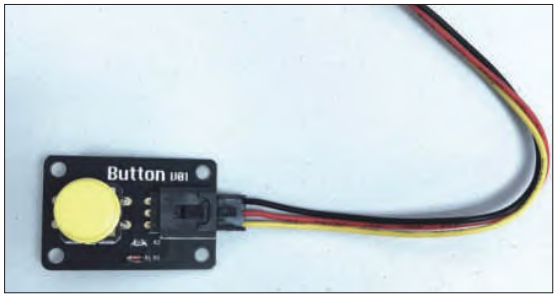


图 3-3 按钮板实物图

灯板：配有专用连接线，线序定义是：黄色为 GPIO，红色为 3.3V，黑色为 GND。灯板实物如图 3-4 所示。

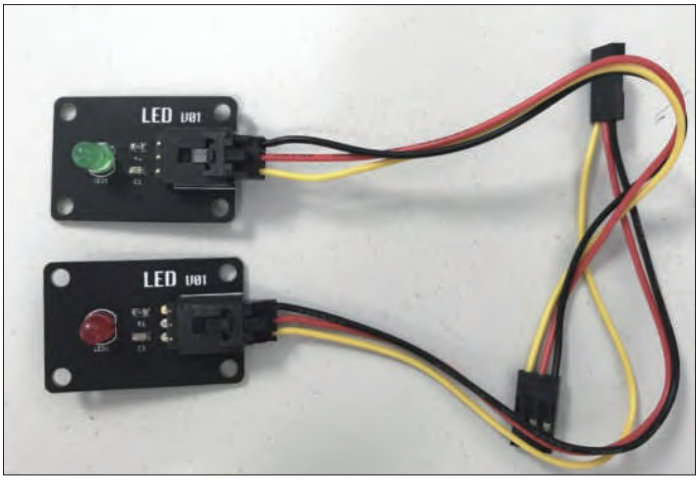


图 3-4　灯板实物图

TTL 线和电源适配器：TTL 线内置了 USB 转串口功能，可以将系统启动等信息通过串口方式从底板输出到我们的计算机上进行调试，通过串口方式访问系统，操作起来就相当于让我们的计算机成为 MR808 开发板的“键盘和显示器”。TTL 线如图 3-5 所示，电源适配器如图 3-6 所示。

图 3-5　TTL 线

Tips

GPIO：通用输入输出接口，Linux 系统支持的一种标准外围接口，可以独立工作或模拟成其他接口。

3.3V：表示供电电压为 3.3V。

GND：地线。

图 3-6 电源适配器

3.4.4 AxWRT 系统

AxWRT 系统完全来自 OpenWrt 社区的发行版“BarrierBreaker”，为了满足教学需要，我们重新进行了软件部分的预制和精简，并且遵循 OpenWrt 社区关于 OpenWrt 的授权协议。在接下来的章节中，我们将结合 AxWRT 系统进行讲解。

Tips

AxWRT 系统资源百度网盘下载地址：http://pan.baidu.com/s/1i5chMD3
AxWRT 系统资源服务器下载地址：http://www.maker1000.com/makerouter/

第4章 智能路由开发环境搭建

“留人不住，醉解兰舟去。一棹碧涛春水路，过尽晓莺啼处。”

本章主要介绍 Linux 相关知识，学习这一部分可以帮助读者快速了解 Linux 操作系统的特性，为开始 OpenWrt 之旅做好准备，希望读者能够熟练掌握本章所介绍的指令。

4.1 小企鹅的力量

Linux 操作系统诞生于 1991 年 10 月 5 日，作为全球厂商与黑客开源协同最伟大的案例，Linux 已经深入到我们生活的每个角落。当你享受手机的便捷时，当你感受网络的浪潮时，其背后是全世界数不尽的 Linux 服务器默默地工作。当然，如果没有 Linux，也不会有 OpenWrt，更不会有今天的智能路由器；如果没有 Linux，就不会有手机上的安卓系统，也没有现在的小米这些厂商。

Linux 到底有多少开发者，估计没有人能说得清楚，OpenWrt 作为一个基于 Linux 的嵌入式发行版，其语法和标准当然都完全符合 Linux 的标准，也因此，在网上跟爱好者互动的时候发现很多爱好者并不十分了解 Linux，以至于学习本书的时候会出现只能照抄的情况。

Linux 的内核为整个系统的核心，内核有专门的网站进行维护，其中大部分都是全球各类顶级厂商，目前中国的公司对内核贡献还比较少。

文件系统：文件系统是操作系统的重要组成部分，所谓文件系统是指操作系统中负责管理和存储文件信息的软件机构。文件系统的存在，使得数据可以被有效而透明地存取、访问。文件系统是指在一个物理设备上的任何文件组织和目录、Linux 程序、库、系统文件和用户文件。

Linux 通过把系统支持的各种文件系统链接到一个单独的树形层次结构中，用来实现对多文件系统的支持。该树形层次结构把文件系统表示成一个整个的独立实体。无论什么类型的文件系统，都被装配到某个目录上，由被装配的文件系统的文件覆盖该目录原有的内容。该目录被称为装配目录或装配点。在文件系统卸载时，装配目录中原有的文件才会显示出来。

Linux 的绝大多数功能都是由文件系统来完成的，文件系统是由几千到上万个独立软件包在一起组成的，可能你所使用的每一个命令都是一个独立软件包，都有独立的公司在负责。

不同的团队和厂商使用内核与文件系统搭配在一起，经过一些修改就产生了不同的发行版。发行版一般是商业称呼，比如 CentOS 系统是发行版，安卓系统是发行版，OpenWrt

系统也是发行版。不论这些系统做得有多么鹤立鸡群，都无法抹去它们是 Linux 不同程度的发行版这一命运。

4.2 准备学习工具

智能路由器亦属于嵌入式设备，Linux 是嵌入式设备上比较流行的操作系统。开发智能路由器首先要构建好一个主机开发环境，嵌入式 Linux 下的大部分开发工作都是在通用 PC 中完成的。这是进行嵌入式 Linux 开发的必经之路。为了操作方便，我们采用虚拟机方式，读者仅需要有 Windows，安装虚拟机即可。

VirtualBox 虚拟机软件，本书推荐使用这款软件，它是一款开源、免费的虚拟机软件，用于学习是非常方便的，请先下载准备好。

Tips

VirtualBox 的官方下载地址：https://www.virtualbox.org/wiki/Downloads
VirtualBox 的百度网盘下载地址：http://pan.baidu.com/s/1mivE4pq
VirtualBox 的服务器下载地址：http://www.maker1000.com/makerouter/software/
主包文件：VirtualBox-4.3.20-96997-Win.exe
扩展包文件：Oracle_VM_VirtualBox_Extension_Pack-4.3.20-96996.vbox-extpack

CentOS 操作系统，是 Linux 的一种发行版，来源自 RedHat 公司的企业级 Linux 克隆版，CentOS 6.6 可以在官方永久地址进行下载。

Tips

CentOS 6.6 的官方永久地址：http://archive.kernel.org/centos-vault/6.6/isos/x86_64/

4.3 安装 VirtualBox 虚拟机

1 双击软件包 VirtualBox-4.3.20-96997-Win.exe 打开安装程序，单击“Next”进入下一步，如图 4-1、图 4-2 所示。

■ 图 4-1 打开安装界面

图 4-2　Next 界面

2 可以修改 VirtualBox 的安装位置或按默认安装，单击“Next”进入下一步，如图 4-3 所示。

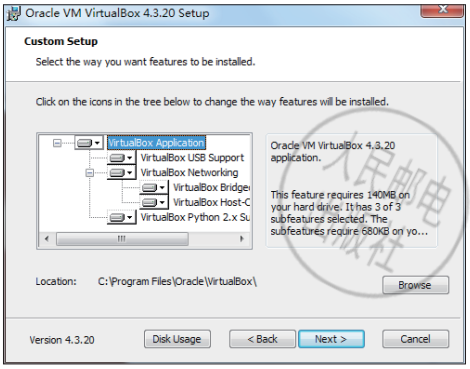


图 4-3　安装位置

3 创建图表选择，默认继续下一步，单击“Next”进入下一步，如图 4-4 所示。

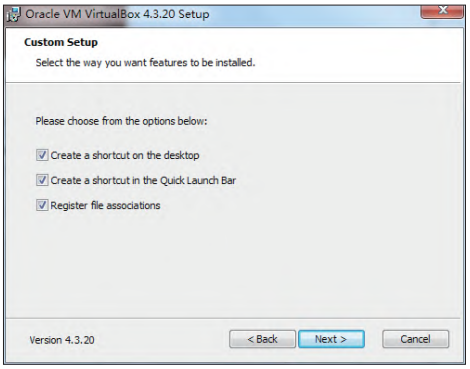
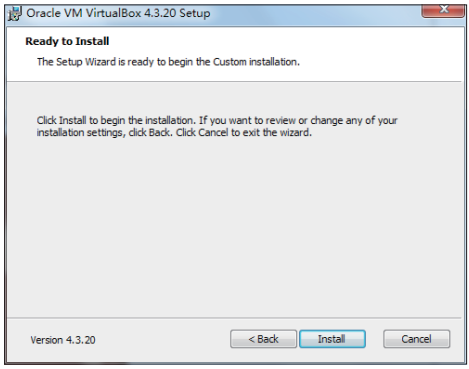


图 4-4　选择创建图标

4 单击“Yes”确认安装，再单击“Install”等待安装完毕，如图 4-5、图 4-6 所示。

■ 图 4-5 单击“Yes”确认安装



■ 图 4-6 单击“Install”安装

5 单击“Finish”确认安装完成，然后将自动开启 VirtualBox 主界面，如图 4-7 所示。

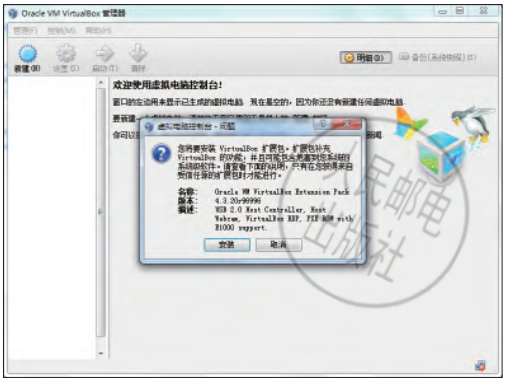
■ 图 4-7 安装完成

注意：VirtualBox 虽然升级到了 4.3.0，但是存在很多问题，比如：（1）无法很好地在 Windows 7 上运行；（2）安装程序后引起 Windows 7 系统的 UDP 广播协议工作出现问题，致使搞开发的工程师在调试网络 UDP 的广播相关程序时无法正常通信。VirtualBox 安装在 Windows 7 后，双击程序运行，出现了“被召者 RC: E_NOINTERFACE（0x80004002）”的错误提示，解决办法是：在 VirtualBox 程序图标上单击鼠标右键→“属性”→“兼容性”，勾选“以兼容模式运行这个程序”，在“特权等级”中勾选“以管理员身份运行此程序”，只要完成这两步，确定后，重新运行程序，就会发现再也没有错误提示，程序跑起来了。

接下来安装 VirtualBox 的扩展包。

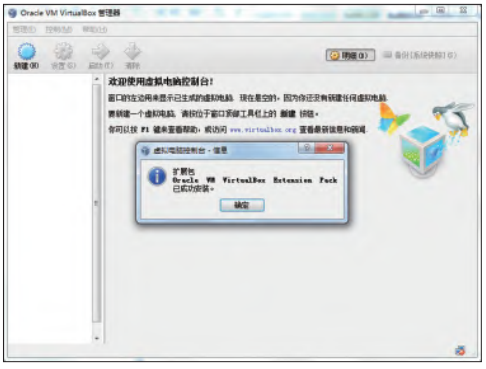
1 双击下载的文件 Oracle_VM_VirtualBox_Extension_Pack-4.3.20-96996.vbox-extpack。

2 VirtualBox 将提示“您将要安装 VirtualBox 扩展包”，单击“安装”，如图 4-8 所示。



■ 图 4-8　安装 VirtualBox 扩展包

3 鼠标滑动查看“VirtualBox 许可”协议，一直到屏幕右下角显示“我同意”，单击它。完成安装，如图 4-9 所示。



■ 图 4-9　扩展包安装完成

4.4 在虚拟机中安装 Centos

4.4.1 创建新虚拟主机

在安装 Centos 之前要确保你的 C 盘有足够的空间（20GB），因为在未来的日子中，我们会用这个系统来教大家如何自己编译 OpenWrt。

1 打开 VirtualBox 的主界面，单击“新建”图标。

2 为你的虚拟机起一个名字，类型选择为 Linux，版本选择为 Red Hat（64bit），单击“下一步”，如图 4-10 所示。

图 4-10 Linux 版本选择

3 为完成后续操作，建议内存至少分配 1024MB，有条件可以分配更大，单击“下一步”，如图 4-11 所示。

图 4-11 内存容量选择

4 现在创建虚拟硬盘，单击“新建”，类型选择为 VDI，单击“下一步”，分配方式选择为“动态分配”，磁盘容量建议设置为 20GB。动态创建的磁盘不会立即占用你硬盘 20GB 空间，会根据需要占用。磁盘选择如图 4-12 所示。

图 4-12 磁盘选择

4.4.2 配置新虚拟机

1 在虚拟机的名字上单击鼠标右键，选择“设置”，如图 4-13 所示。

图 4-13 虚拟机设置界面

2 在设置窗口左侧单击“系统”，在右侧“主板”选项卡的“扩展特性”下勾选“启用 I/O APIC”，如图 4-14 所示。

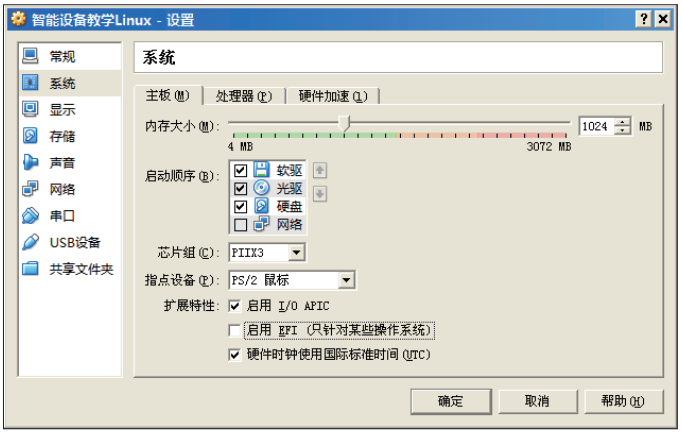


图 4-14 系统主板配置界面

3 在右侧单击“处理器”选项卡，在绿色区域尽量多地选择 CPU 内核，选择得越多，未来我们编译系统的时候越节省时间，如图 4-15 所示。

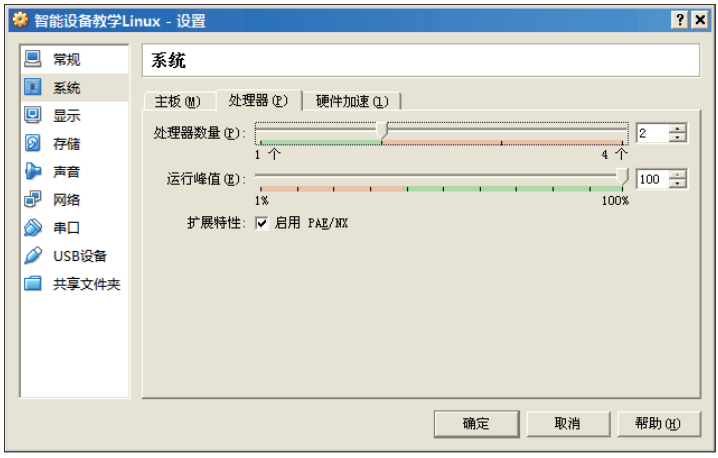


图 4-15 处理器配置界面

4 在左侧选择“网络”，右侧选项卡显示出“网卡 1”，选择网卡 1 的连接方式，如果选择“桥接网卡”，则表示虚拟机通过计算机的网卡连接外网，但是有独立的 IP 地址，推荐使用这个模式。在桥接网卡模式下，还要在界面名称中选择你的计算机上正在使用的网卡是哪个。网卡配置界面如图 4-16 所示。

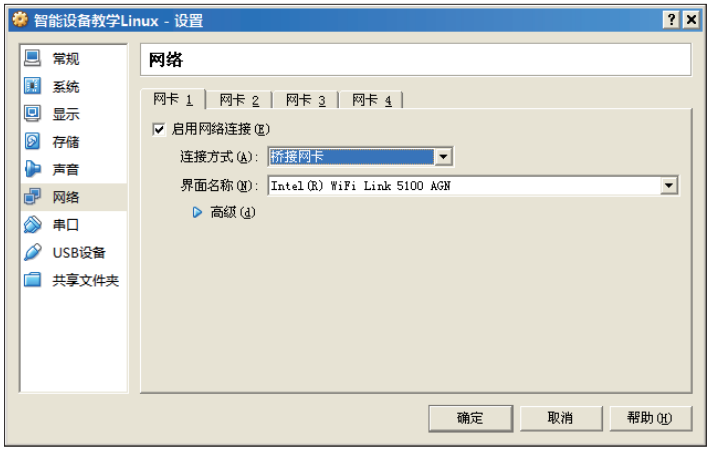


图 4-16 网卡配置界面

5 在设置窗口左侧选择“存储”，在中间选择没有盘片的图标（表示光驱），在右侧单击光盘图标，选择下载的 CentOS-6.6-x86_64-minimal.iso 文件。存储配置界面如图 4-17 所示。

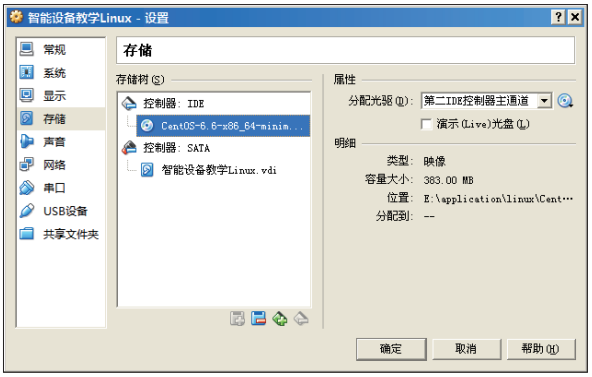


图 4-17　存储配置界面

6 单击“确定”，关闭设置窗口，这样当下次启动的时候，系统就会进入安装光盘状态了。

4.4.3　安装 Centos 系统

VirtualBox 虚拟机使用起来就像本地计算机一样，但是要注意捕捉模式，当单击虚拟机窗口后会进入捕捉模式（此时鼠标和键盘操作都将只会在虚拟机窗口中反应）。如果要结束捕捉模式，请按右 Ctrl 键。这非常重要，请切记！

1 在虚拟机列表中选中刚才所创建的那个虚拟机，单击启动图标，可以进入 Centos 安装界面，如图 4-18 所示。

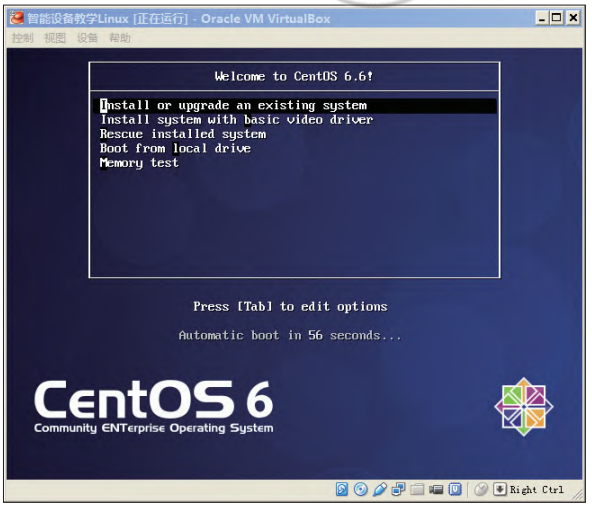
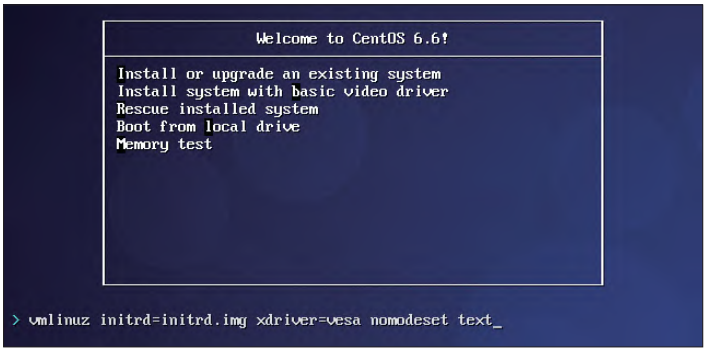


图 4-18　Centos 安装界面

2 为了加快安装进度，我们采用文字安装界面。选中“Install system with basic video driver”，然后按键盘上的 Tab 键，将进入编辑模式。在出现的编辑窗口输入“空格

text”，就像图 4-19 所示那样，然后按回车键。



■ 图 4-19 选择文字安装

3 在看到一堆启动信息后，正式进入了安装界面，第一个页面是提示进行光盘检测的，我们跳过去就可以了，选择“Skip”，按回车键。

4 看到“Welcome to CentOS!”后单击“OK”。后续操作用键盘的上下左右键可以选择，用 Tab 键可以在不同的选项间跳动，用回车键可以实现确认，用退出键可以取消，请切记。

5 在“Language Selection”中选择语言，强烈推荐使用默认的英文，不要选择中文，不然会造成很多误解，尤其是诸多专业词语的中文翻译并不统一。用键盘上下键选择“English”，按回车键确认。

6 在“Keyboard Selection”中选择键盘类型，如果是正常键盘，选择“us”，按回车键确认。

7 存储器检测，可能会提示“Warning”（警告），发现了一个存储器是否可以使用。那就是系统以为虚拟机的磁盘有东西，没关系，用 Tab 键跳到 Re-initialize 上，按回车键继续。

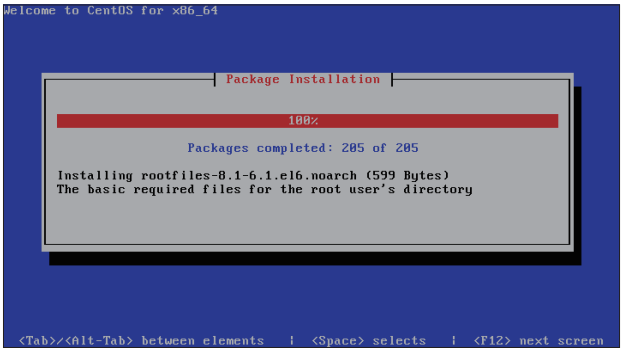
8 在“Time Zone Selection”中选择时区，这就不要随便选了，用键盘上下选到“Asia/Shanghai”，按 Tab 键跳到“OK”，按回车键确认。

9 在“Root Password”中设置超级用户密码，root 是整个系统的超级用户，这时请设置一个至少 8 位、不要过于简单的密码，密码要填写两次，第一次填写在 Password 中，第二次填写在 Password（confirm）中，两次一致后，按 Tab 键跳到“OK”，按回车键确认。如果系统提示“Weak Password”，表示你的密码太简单了，请“Cancel”，重新来过吧。

10 在“Partitioning Type”中选择分区方法，这个时候，选择“Use entire drive”没错，使用整个虚拟磁盘安装系统，下面显示的 sda 表示当前的第一个虚拟磁盘，按 Tab 键跳到“OK”，按回车键确认。

11 “Writing storage configuration to disk”，最后一次提示是否开始写虚拟磁盘，别等它废话了，按 Tab 键跳到“Write changes to disk”，按回车键就对了。

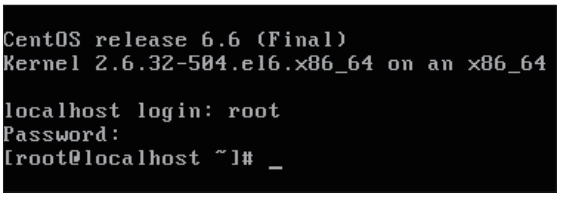
12 屏幕中将跳出一堆信息，进入安装流程，请耐心等待 10~30 分钟。安装进度如图 4-20 所示。



■ 图 4-20　安装进度

13 经过漫长的等待，终于出现“Complete”的时候，选择 “Reboot”并按回车键。VirtualBox 会很 Nice 地自动把虚拟光盘卸载掉，所以不用怕再次进入安装界面。等待进度条读完，就会显示登录界面了。

14 在登录界面中输入账号“root”并按回车，然后输入密码并按回车键（输入密码时是不显示的），回车后如果输入的密码没错，就会显示提示符。账号登录界面如图 4-21 所示。



■ 图 4-21　账号登录

15 输入以下指令，即可以立即执行关机流程。

```
# 输入以下指令，系统将执行关机流程
[root@localhost ~]# shutdown -h now
```

上面的内容为指令演示，其中 [root@localhost ~]# 后面跟随的内容为输入的指令，以 # 开头的表示注释内容，仅为了讲解，不是指令内容，以后所有章节都采用这种方式演示指令，不再另外说明。

Tips

root：Linux 系统的超级用户，拥有整个系统的全部权限，在本书中为了简单，一律使用该账号做演示。请注意 root 必须为全部小写。

到此为止，本节内容完成，如果在操作过程中出现错误导致操作失败，请参考以下做法。

- 虚拟机创建过程失败：在 VirtualBox 主界面右键选择虚拟机，选择“删除”，重新创建。
- 重新安装 Centos：将旧的删除掉，然后重新创建即可。

4.5 SSH 远程登录

4.5.1 SSH 的作用

SSH 的全称为 Secure Shell，是一种安全的网络远程登录系统，而 Shell 的意思就是命令行模式，我们在启动虚拟机下安装的系统完毕后会进入一个黑色屏幕，上面有字幕提示的界面，就是命令行的登录界面。SSH 可以让使用者通过网络，远程地登录所安装的虚拟机。也因此，SSH 同样适用于 OpenWrt 系统的远程登录。

4.5.2 启动网卡

默认的精简版 CentOS 没有启动网卡，需要手工修改为启动模式，请先启动 CentOS 虚拟机，在虚拟机的窗口中进行如下操作。

1 首先登录系统，填写账号，然后填写密码，完成系统登录，进入命令行模式。

2 手工输入启动网卡指令。

```
# 临时启动 eth0 网卡
[root@localhost ~]# ifup eth0
```

3 如果想每次启动机器都可以启动网卡，有个简单的办法，直接将启动网卡写入启动选项里。

```
# 将启动网卡写入启动选项文件中（该指令没有返回值）
[root@localhost ~]# echo "ifup eth0 &" >> /etc/rc.local
```

4 查看是否写入成功。

```
[root@localhost ~]# cat /etc/rc.loal
#!/bin/sh
#
```

```
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
ifup eth0 &
```

4.5.3 查看当前 IP

如果你的网络中没有 DHCP 服务器虚拟机，获得不到 IP，请先参考上一章内容，解决网络部分的问题。输入以下指令可以获知当前机器的 IP 地址。

```
# 指令输入后显示的“inet addr:”后面的内容即当前这台虚拟机的 IP 地址
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr XX:XX:XX:XX:XX:XX
          inet addr:192.168.199.207  Bcast:192.168.199.255
Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feae:ce41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:311 errors:0 dropped:0 overruns:0 frame:0
          TX packets:110 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34524 (33.7 KiB)  TX bytes:13174 (12.8 KiB)
```

屏幕中所显示的“inet addr:192.168.199.207”即本书虚拟机 IP 地址，不要认为 192.168.199.207 就是你们虚拟机的 IP 地址，要知道你们的虚拟机的 IP 地址是从你们的路由器申请到的，肯定是不一样的，所以千万别直接复制。

4.5.4 SSH 连接

不知道读者是不是感觉到了，用虚拟机的窗口直接操作系统还是不太方便，现在我们可以开始安装 SSH 终端软件，然后通过 SSH 连接系统进行操作，比用虚拟机窗口方便很多，当然虚拟机窗口是不能关闭的，一旦关闭了，虚拟机就挂了！

本书使用 PuTTY 串口终端软件，PuTTY 是一个特别好用的轻量级 SSH 客户端。下载地址如下。

Tips

PuTTY 下载地址：http://pan.baidu.com/s/1mhAXvmS
PuTTY 的服务器下载地址：http://www.maker1000.com/makerouter/software/putty.exe
PuTTY 官方网站：http://www.putty.org/

1 PuTTY 不用安装，直接执行就可以了。执行后填写连接信息，PuTTY 的配置如图 4-22 所示。

（1）在 Host Name 中填写刚才虚拟机的 IP 地址。

（2）Connection Type（连接类型）选择 SSH。

（3）Port（端口）填写 22。

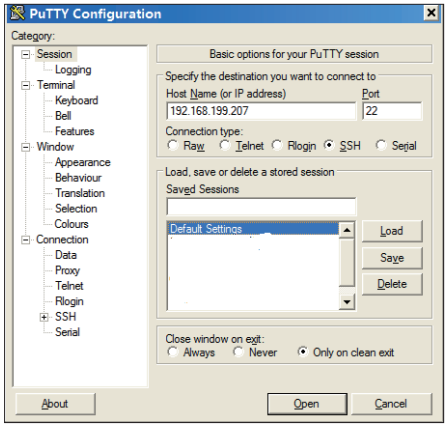


图 4-22 PuTTY 配置界面

2 单击“Open”进行连接。第一次连接会提示“PuTTY Security Alert”，说的是服务端密钥没有被存储，是否信赖这个服务端，直接单击“是”就行了。

3 在屏幕中提示“login as:”后开始填写账号，账号就是 root，密码跟使用虚拟机窗口登录所填写的密码一样，如果登录成功，就会显示提示符。如果长时间未填写或多次失败，将提示错误，断开连接，遇到此情况，重新连接即可。

```
# 本机屏幕显示的效果
login as: root
root@192.168.199.207's password:
Last login: Tue Jan 20 11:10:32 2015 from china-811071111.lan
[root@localhost ~]#
```

PuTTY 长时间未操作可能会出现断线的情况，这时只要重新连接即可。SSH 是 Linux 下最常用的工具了，远程连接十分方便，也为我们使用 OpenWrt 做好准备，因为 OpenWrt 所运行的设备往往都没有显示器，这时使用 SSH 远程连接成为了一种技术常态。

4.6 WinSCP 文件传输

4.6.1 SCP 协议

SCP（Security Copy）协议是 SSH 协议的一个子系统，用于实现通过 SSH 协议的文件传输。另外还有一种 SFTP 协议，SFTP 协议在 OpenWrt 中默认没有被启用，有 SCP 足矣。SCP 简单方便又好用。

4.6.2 安装 WinSCP 客户端

WinSCP 是一个开源免费软件，在你的计算机中安装 WinSCP 客户端吧，这是一个强大的工具，可以实现 SCP 文件的上传和下载。这个工具的下载地址已经准备好了。

Tips

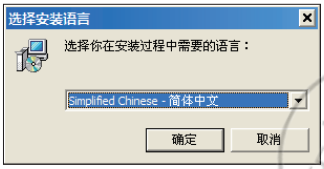
WinSCP 下载地址：http://pan.baidu.com/s/1pK732C7

WinSCP 的服务器载地址：http://www.maker1000.com/makerouter/software/winscp439setup.zip

WinSCP 官网：http://winscp.net/

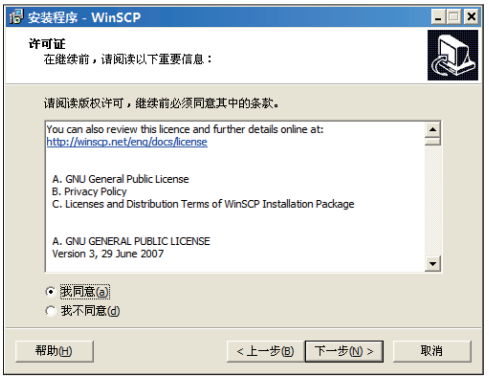
1 解压压缩包。

2 双击 winscpsetup.exe 文件，选择语言界面如图 4-23 所示。



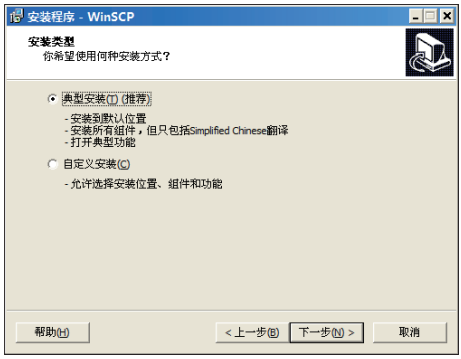
■ 图 4-23 选择语言

3 单击“确定”，然后同意许可协议，单击“下一步”，如图 4-24 所示。



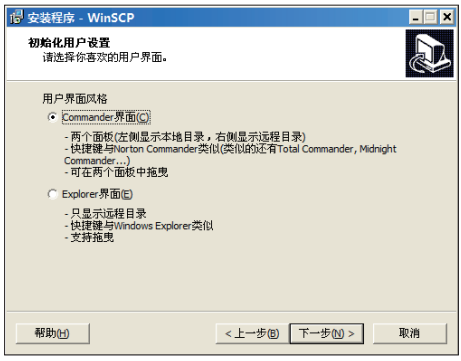
■ 图 4-24 同意许可协议

4 安装类型选择“典型安装”，单击“下一步”，如图 4-25 所示。



■ 图 4-25　安装类型选择

5 界面风格选择“Commander”，单击“下一步”，如图 4-26 所示。



■ 图 4-26　界面风格选择

6 单击“安装”后完成安装流程，如图 4-27 所示。

■ 图 4-27　安装确认

4.6.3 创建到 Linux 的连接

1 还记得你的虚拟机的 IP 地址吗？如果不记得了，请查看之前的章节。打开 WinSCP 的主界面。

2 单击屏幕右上侧的“新建”，配置界面如图 4-28 所示。

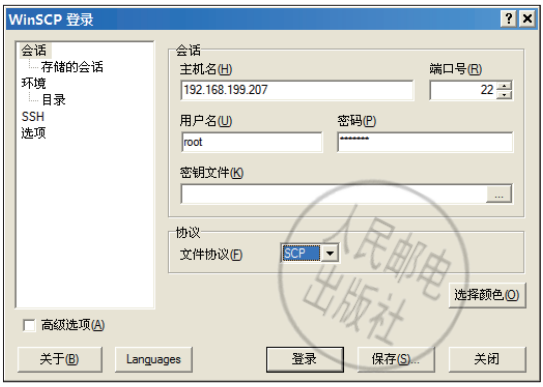
（1）在主机名中填写虚拟机的 IP 地址。

（2）端口号填写 22。

（3）在用户名中填写超级用户 root。

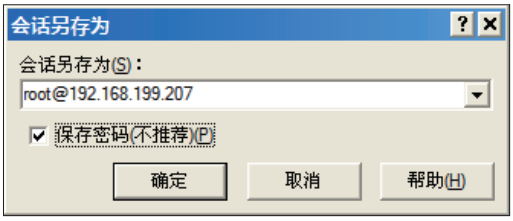
（4）在密码中填写你设置的密码。

（5）文件协议一定要选择 SCP。



■ 图 4-28 WinSCP 配置界面

3 单击“保存”，弹出保存对话框，选择“保存密码”，省得下次再输入了，如图 4-29 所示。



■ 图 4-29 保存会话

4 再次返回登录对话框。这时选中你刚才创建的那个连接。单击屏幕下角的“登录”。首次连接可能会有警告，选“是”即可。

5 如果信息没填写错，就可以登录到主界面了。主界面左侧为本地磁盘，右侧就是虚拟机上的磁盘了。

4.6.4 文件传输

WinSCP 中的文件传输非常容易操作，操作方法如下。

- 向 Linux 上传文件：在左侧选中文件，拖动到右侧即可以完成上传。
- 从 Linux 下载文件：在右侧选中文件，拖动到左侧即可完成下载。
- 删除 Linux 中的文件：在右侧选中文件，按键盘上的 Delete 键就会提示是否删除。
- 刷新文件列表：WinSCP 不会自动刷新 Linux 上的文件列表，如果有变动是看不到的，单击屏幕右上方的刷新按钮即可完成刷新，如图 4-30 所示。

图 4-30 刷新按钮

第5章 基本知识

5.1 向系统下达命令

本章所要介绍的内容是迄今为止最多的一章，没有接触过 Linux 的读者，难度较大，如果遇到困难，可以多上网查阅参考资料。本章所介绍指令都是 OpenWrt 系统下的常用指令，作为一个智能设备的专业人士，这些都是必备的知识。

现在你既可以通过 SSH，也可以通过虚拟机的窗口向系统下达命令。指令一般由 3 部分组成。

```
[root@localhost ~]# ifup eth0
```

以上指令语法和含义说明如下。

语法格式：提示符 命令 参数 1 参数 2……

- 常见的命令大部分是由 3 部分组成的，其中参数是可选的。
- 举例的 [root@localhost ~]# 这一部分是提示符，由操作系统显示出来。
- 举例的 ifup 这一部分是命令。
- 举例的 eth0 这一部分是参数，参数可能有多个。

命令输入完成后按回车键即可被执行。如果所输入的命令有错，用组合键 Ctrl+C 可以中断输入，或中断执行。

在 Linux 系统下是区分大小写的，大部分 Linux 的命令都是小写的，请注意大小写。

5.2 时间命令

5.2.1 查看当前日期

语法格式：date

```
[root@localhost ~]# date
Tue Jan 20 13:07:41 CST 2015
```

5.2.2 查看当前日历

语法格式：cal

```
[root@localhost ~]# cal
    January 2015
Su Mo Tu We Th Fr Sa
            1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

5.3 关机和重启

5.3.1 立即安全关闭系统

语法格式：shutdown 参数……

```
[root@localhost ~]# shutdown -h now
```

这个指令执行后会提示信息，然后关机，在嵌入式设备上不会关机，只会停止运行系统。

5.3.2 重新启动系统

语法格式：reboot

```
[root@localhost ~]# reboot
```

这个指令执行后会提示信息，然后重新启动。

5.3.3 强制将缓存写入存储器

语法格式：sync

```
[root@localhost ~]# sync
```

这个指令用来保证在重启之后不会因缓存没有保存到磁盘上而导致文件损坏。这个指令没有任何返回信息。

5.4 权限机制

在 Linux 系统下所有文件和目录（在 Windows 下叫作文件夹）都具有一定的权限范围，并且有严格的限制。

```
# 执行 ls -l 指令获得当前文件夹中的一般文件
[root@localhost ~]# ls -l
total 20
-rw-------. 1 root root 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-r--r--. 1 root root 8823 Jan 18 18:51 install.log
-rw-r--r--. 1 root root 3314 Jan 18 18:43 install.log.syslog
```

5.4.1 文件权限

install.log 举例为 **-rw-r--r--.** 1 root root 8823 Jan 18 18:51 install.log

- 第一个字符表示当前是否是文件夹，如果显示为 d 表示目录，如果显示为 -，表示非目录。
- 接下来每 3 个字符为一组，第一组表示文件所有者自身对该文件的权限，第二组表示相同用户组对该文件的权限，第三组表示其他人对该文件的权限。以 - 表示不存在权限，权限用 r（可读）、w（可写）、x（可执行）3 个字符表示。
- install 的权限设置表明，该文件非目录，文件所有者具备对文件的可读、可写权限，同组和其他人只可读。

5.4.2 所有者

install.log 举例为 -rw-r--r--. 1 **root root** 8823 Jan 18 18:51 install.log

- 第一个 root 所在的位置，表示这个文件的所有者用户。
- 第二个 root 表示这个文件所有者的用户分组。
- 对应文件权限，可以详细地分配、限制每个文件都是归谁所有的。

5.4.3 文件容量

install.log 举例为 -rw-r--r--. 1 root root **8823** Jan 18 18:51 install.log

- 数字 8823 表示该文件的容量，是以字节为单位表示的，如果要换算成 KB，要用该数值除以 1024；如果要算成 MB，要用该数值除以 1024 两次。
- install.log 这个文件的大小约是 8.6KB。

5.4.4 文件日期

install.log 举例为 -rw-r--r--. 1 root root 8823 **Jan 18 18:51 install.log**

- Jan 18 18:51 为文件的修改日期。
- 文件名是最后一个被显示出来的。

5.4.5 可执行文件

Linux 与 Windows 不同，在 Windows 中可执行文件以 .exe 作为文件名的结尾。在 Linux 中，可执行文件主要看以下参数：

- 文件是否被给予了执行权限；
- 是二进制可执行文件，还是脚本程序。

5.4.6 隐含文件

Linux 中隐含文件或隐含目录以 . 开头表示。这样可以查看到。

```
[root@localhost ~]# ls -al
total 52
dr-xr-x---.  2 root root 4096 Jan 18 19:01 .
dr-xr-xr-x. 22 root root 4096 Jan 20 13:29 ..
-rw-------.  1 root root 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------.  1 root root  583 Jan 20 13:27 .bash_history
-rw-r--r--.  1 root root   18 May 20  2009 .bash_logout
-rw-r--r--.  1 root root  176 May 20  2009 .bash_profile
-rw-r--r--.  1 root root  176 Sep 23  2004 .bashrc
-rw-r--r--.  1 root root  100 Sep 23  2004 .cshrc
-rw-r--r--.  1 root root 8823 Jan 18 18:51 install.log
-rw-r--r--.  1 root root 3314 Jan 18 18:43 install.log.syslog
-rw-r--r--.  1 root root  129 Dec  4  2004 .tcshrc
```

在相同的文件夹下，可以看到很多以 . 开头的文件和文件夹，这些都是隐含起来的。隐含的目的不是为了藏起来让你找不到，而是因为这些文件都是数据文件，看着让人烦，所以藏起来清净一下。

5.4.7 修改所有人

语法格式：chown [账户] 文件或目录名

只有 root 用户才能将文件所有人修改成为其他人。

```
[root@localhost ~]# chown nobody install.log
[root@localhost ~]# ls -l install.log
-rw-r--r--. 1 nobody root 8823 Jan 18 18:51 install.log
```

修改目录和目录下所有文件。

```
[root@localhost ~]# chown -R nobody ./*
[root@localhost ~]# ls -l
total 20
-rw-------. 1 nobody root 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-r--r--. 1 nobody root 8823 Jan 18 18:51 install.log
-rw-r--r--. 1 nobody root 3314 Jan 18 18:43 install.log.syslog
```

5.4.8 修改所有组

语法格式：chgrp [账户] 文件或目录名

```
[root@localhost ~]# chgrp nobody install.log
[root@localhost ~]# ls -l install.log
-rw-r--r--. 1 root nobody 8823 Jan 18 18:51 install.log
```

修改目录和目录下所有文件。

```
[root@localhost ~]# chgrp -R nobody ./*
[root@localhost ~]# ls -l
total 20
-rw-------. 1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-r--r--. 1 root nobody 8823 Jan 18 18:51 install.log
-rw-r--r--. 1 root nobody 3314 Jan 18 18:43 install.log.syslog
```

5.4.9 修改文件权限

语法格式：chmod 权限编号 文件或目录

- 权限编号：r 为 4，w 为 2，x 为 1。
- 权限编号一共 3 位，表示所有者、所有组、其他人的权限。比如权限编号 760 表示所有者拥有 rwx 权限，所有组拥有读写权限，其他人无权限。

设置 install.log 为 760 权限。

```
[root@localhost ~]# chmod 760 install.log
[root@localhost ~]# ls -l install.log
-rwxrw----. 1 root nobody 8823 Jan 18 18:51 install.log
```

设置 /var/log 文件夹下所有 .log 为结尾的文件为 600 权限。

```
[root@localhost ~]# chmod -R 600 /var/log/*.log
[root@localhost ~]# ls -l /var/log/*.log
-rw-------. 1 root root   2368 Jan 18 18:51 /var/log/anaconda.ifcfg.log
-rw-------. 1 root root  21487 Jan 18 18:51 /var/log/anaconda.log
-rw-------. 1 root root  28841 Jan 18 18:51 /var/log/anaconda.program.log
-rw-------. 1 root root 121087 Jan 18 18:51 /var/log/anaconda.storage.log
-rw-------. 1 root root   2600 Jan 18 18:51 /var/log/anaconda.yum.log
-rw-------. 1 root root   1462 Jan 20 13:29 /var/log/boot.log
-rw-------. 1 root root 172790 Jan 18 18:51 /var/log/dracut.log
```

5.5 文件与目录管理

5.5.1 路径概念

- 根目录：系统的基础目录，以反斜杠 / 表示根目录，Linux 没有 C、D 这种盘符概念，所有的资源都是在根目录之上存在的。

- 绝对路径：从根目录开始写的叫绝对路径，比如 /var/log。
- 相对路径：从当前目录开始找的叫当前路径，当前路径根据你所在的目录位置不同而各不相同，比如你现在在 /root 下面，那么当前路径 ../home 实际上与 /home 这个绝对路径含义是相同的。

5.5.2 特殊目录

```
[root@localhost ~]# ls -al
total 52
dr-xr-x---.  2 root root   4096 Jan 18 19:01 .
dr-xr-xr-x. 22 root root   4096 Jan 20 13:29 ..
-rw-------.  1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------.  1 root root    583 Jan 20 13:27 .bash_history
-rw-r--r--.  1 root root     18 May 20  2009 .bash_logout
-rw-r--r--.  1 root root    176 May 20  2009 .bash_profile
-rw-r--r--.  1 root root    176 Sep 23  2004 .bashrc
-rw-r--r--.  1 root root    100 Sep 23  2004 .cshrc
-rw-------.  1 root nobody 8823 Jan 18 18:51 install.log
-rw-------.  1 root nobody 3314 Jan 18 18:43 install.log.syslog
-rw-r--r--.  1 root root    129 Dec  4  2004 .tcshrc
```

在这里，我们可以看到有两个目录叫作 . 和 .. 。这两个是特殊目录，分别代表特别含义。

- . 代表当前目录，比如你在 /root 下面执行 ls ./ 与你执行 ls /root 是一个意思。
- .. 代表上一级目录，比如你在 /root 下面执行 ls ../ 与你执行 ls / 是一个意思。
- 这两个特殊目录都是方便用来作相对路径表达的。

5.5.3 变换目录

语法格式：cd [相对路径或绝对路径]

```
[root@localhost ~]# cd /tmp/
[root@localhost tmp]# ls
yum.log
[root@localhost tmp]# cd /root/
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
```

5.5.4 查看当前所在目录是哪个

语法格式：pwd

```
[root@localhost ~]# pwd
/root
```

5.5.5 创建目录

语法格式：mkdir [新目录名字]

```
[root@localhost ~]# mkdir test
[root@localhost ~]# ls -l
total 24
-rw-------. 1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------. 1 root nobody 8823 Jan 18 18:51 install.log
-rw-------. 1 root nobody 3314 Jan 18 18:43 install.log.syslog
drwxr-xr-x. 2 root root   4096 Jan 20 14:24 test
```

5.5.6 删除目录

语法格式：rmdir [目录名字]

```
[root@localhost ~]# rmdir /root/test/
[root@localhost ~]# ls -l /root/
total 20
-rw-------. 1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------. 1 root nobody 8823 Jan 18 18:51 install.log
-rw-------. 1 root nobody 3314 Jan 18 18:43 install.log.syslog
```

删除目录指令只能删除掉空的目录，如果目录下面还有其他文件和目录，就要使用 rm 命令了。

5.5.7 列表文件和目录

语法格式：ls [目录或文件]

显示当前路径下的文件。

```
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
```

显示文件详细信息。

```
[root@localhost ~]# ls -l
total 20
-rw-------. 1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------. 1 root nobody 8823 Jan 18 18:51 install.log
-rw-------. 1 root nobody 3314 Jan 18 18:43 install.log.syslog
```

显示包括隐含文件。

```
root@localhost ~]# ls -al
```

```
total 52
dr-xr-x---.  2 root root   4096 Jan 20 14:25 .
dr-xr-xr-x. 22 root root   4096 Jan 20 13:29 ..
-rw-------.  1 root nobody 1104 Jan 18 18:51 anaconda-ks.cfg
-rw-------.  1 root root    583 Jan 20 13:27 .bash_history
-rw-r--r--.  1 root root     18 May 20  2009 .bash_logout
-rw-r--r--.  1 root root    176 May 20  2009 .bash_profile
-rw-r--r--.  1 root root    176 Sep 23  2004 .bashrc
-rw-r--r--.  1 root root    100 Sep 23  2004 .cshrc
-rw-------.  1 root nobody 8823 Jan 18 18:51 install.log
-rw-------.  1 root nobody 3314 Jan 18 18:43 install.log.syslog
-rw-r--r--.  1 root root    129 Dec  4  2004 .tcshrc
```

显示其他路径下的文件详细信息。

```
[root@localhost ~]# ls -l /var
total 60
drwxr-xr-x.  4 root root 4096 Jan 18 18:38 cache
drwxr-xr-x.  3 root root 4096 Jan 18 18:43 db
drwxr-xr-x.  3 root root 4096 Jan 18 18:43 empty
drwxr-xr-x.  2 root root 4096 Sep 23  2011 games
drwxr-xr-x. 15 root root 4096 Jan 18 18:43 lib
drwxr-xr-x.  2 root root 4096 Sep 23  2011 local
drwxrwxr-x.  5 root lock 4096 Jan 18 18:43 lock
drwxr-xr-x.  3 root root 4096 Jan 20 13:29 log
lrwxrwxrwx.  1 root root   10 Jan 18 18:37 mail -> spool/mail
drwxr-xr-x.  2 root root 4096 Sep 23  2011 nis
drwxr-xr-x.  2 root root 4096 Sep 23  2011 opt
drwxr-xr-x.  2 root root 4096 Sep 23  2011 preserve
drwxr-xr-x. 11 root root 4096 Jan 20 13:29 run
drwxr-xr-x.  8 root root 4096 Jan 18 18:40 spool
drwxrwxrwt.  2 root root 4096 Jan 18 18:51 tmp
drwxr-xr-x.  2 root root 4096 Sep 23  2011 yp
```

5.5.8 复制文件或目录

语法格式：cp [参数] 原始位置 目标位置

常用参数：

- -a 保留原有权限。
- -af 保持原有权限，复制原始位置下的所有文件及目录。
- -avf 保留原有权限，复制原始位置下的所有文件及目录，并且在屏幕上显示过程。

复制文件。

```
[root@localhost ~]# cp -av /var/log/messages ./
`/var/log/messages' -> `./messages'
```

```
[root@localhost ~]# ls messages
messages
```

复制目录。

```
[root@localhost ~]# cp -avf /tmp/ ./
`/tmp/' -> `./tmp'
`/tmp/.ICE-unix' -> `./tmp/.ICE-unix'
`/tmp/yum.log' -> `./tmp/yum.log'
[root@localhost ~]# ls -l tmp/
total 0
-rw-------. 1 root root 0 Jan 18 18:37 yum.log
[root@localhost ~]# ls -al tmp/
total 12
drwxrwxrwt. 3 root root 4096 Jan 20 13:29 .
dr-xr-x---. 3 root root 4096 Jan 20 14:40 ..
drwxrwxrwt. 2 root root 4096 Jan 20 13:29 .ICE-unix
-rw-------. 1 root root    0 Jan 18 18:37 yum.log
```

5.5.9 删除文件或整个目录

- 语法格式：rm [参数] 路径或文件名

  常用参数：

- -f 强制删除，不提示确认。
- -r 递归删除，还记得 rmdir 只能删空目录吗？用这个参数就可以把整个目录删除掉。
- 注意：如果你不小心写了个“rm -rf /”就悲剧了，整个操作系统都被你删除了！！！


```
[root@localhost ~]# rm -f messages
[root@localhost ~]# ls messages
```

```
[root@localhost ~]# rm -rf /root/tmp/
[root@localhost ~]# ls ./tmp/
ls: cannot access ./tmp/: No such file or directory
```

5.5.10 移动或更名

语法格式：mv [参数] 原始名字或路径 新名字或路径

常用参数：

- -f 强制完成。

对文件进行更名。

```
[root@localhost ~]# mv install.log abc.log
[root@localhost ~]# ls abc.log
```

```
abc.log
```

对文件进行移动。

```
[root@localhost ~]# mv ./abc.log /tmp/
[root@localhost ~]# ls /tmp/abc.log
/tmp/abc.log
```

5.6 文件分析

文件分析语法也是系统中最常用的，我们经常要查看各种文件中的内容。

5.6.1 查看文件内容

语法格式：cat [文件名]

cat 指令仅限于查看文本文件。

```
[root@localhost ~]# cat anaconda-ks.cfg
```

5.6.2 分行查看

语法格式：cat [文件名]|more

正常使用 cat 会发现内容太多，一下子就过去了，没看清楚吧？ more 命令可以实现分行查看。

在查看过程中会显示一页，然后按回车键一行一行地弹出，一直到文件显示完成。结束分行查看的方法是按键盘上的组合键 Ctrl+C。

```
[root@localhost ~]# cat anaconda-ks.cfg |more
```

5.6.3 查看文件最后几行

语法格式：tail [参数] 文件

常用参数：

- -n [行数]，比如 -n 50 表示 50 行。

有时候，分行查看也并不适合，可能你只想看文件最后 50 行，但是文件有 100MB 那么大，怎么看都不合适，若用 tail 最合适了。

```
[root@localhost ~]# tail -n 50 anaconda-ks.cfg
```

5.6.4 查看启动信息

语法格式：dmesg

系统启动后的启动信息可以用来知道当前系统是否正常启动，这个指令用于解决设备或系统的一些故障信息分析。使用 dmesg 就可以查看系统启动信息。

```
[root@localhost ~]# dmesg
```

也可以配合 more 语法分行查看。

```
[root@localhost ~]# dmesg|more
```

5.7 更多指令

还有一些指令也是本书未来需要使用的，但使用次数不是很多，这里介绍指令的名字，希望读者能够自己学会这些指令。

- 查看进程：ps
- 查看内存：free
- 查看内核版本：uname -r
- 查看工作时间：uptime
- 查看网络端口：netstat -lnp
- 查看磁盘使用情况：df
- 查看已经装载的分区：mount
- 查看系统可用磁盘和分区：fdisk -l

5.8 Vi 编辑器

5.8.1 Vi 介绍

Vi 是全世界所有 Linux 系统标配的编辑器，在 Vi 被发明的时候，计算机还没有小键盘。Vi 是非常非常强大的编辑器，在这里，我们只需要使用 Vi 最基本的指令就可以实现对各种配置文件的编辑。

5.8.2 Vi 的命令

语法格式：vi [文件名]

如果所填写的文件名是存在的，则编辑该文件，如果不存在，就表示创建一个新文件。

```
[root@localhost ~]# vi test
```

则屏幕会显示大概如下内容。

```
~
~
~
~
~
~
~
~
~
~
"test" [New File]
```

屏幕下面显示“New File”表示当前这个文件不存在。下面我们介绍一下 Vi 的使用方法，先不要着急来修改这个文件。

5.8.3 Vi 的 3 个状态

Command mode **命令行模式**：这是打开文件的默认状态，在这个状态下可以通过箭头按键移动查看文件的全部内容。如果是新文件，当然就移动不了了。

- 在命令行模式下可以进入另外两种模式。
- 在命令行模式可以删除某一行内容，将光标移动到该行上快速敲击键盘 d 按键两次。

Insert mode **插入模式**：在这个模式下，可以修改文件内容，或删除文件某个字符。

- 在命令行模式下输入 i 后，将进入插入模式。
- 在插入模式下，按键盘 ESC 键可返回命令行模式。

Last line mode **底行模式**：在这个模式下，可以实现保存文件或退出。

- 底行模式一定要在命令行模式下进入。
- 输入“:w” 保存文件。
- 输入“:wq ”保存文件并且退出。
- 输入“:q! ”退出，但是对文件的改变不做保存。

5.8.4 Vi 编辑操作

创建一个新文件。

```
[root@localhost ~]# vi test
```

屏幕显示。

```
~
```

```
~
~
~
~
~
~
~
~
"test" [New File]
```

输入 i 进入插入模式并且写入 3 行内容。

```
1111111
2222222
3333333
~
~
~
~
~
~
~
-- INSERT --
```


按 ESC 键返回命令行模式。

```
1111111
2222222
3333333
~
~
~
~
~
~
~
```

在命令行模式下，将光标移动到第二行并快速敲击 d 键两次，删除第二行。

```
1111111
3333333
~
~
~
~
~
```

```
~
~
~
```

在命令行模式输入“:wq”并回车，对文件保存并且退出。

```
1111111
3333333
~
~
~
~
~
~
~
~
:wq
```

使用 cat 命令查看一下文件对不对。

```
[root@localhost ~]# cat test
1111111
3333333
```

在编辑文件的过程中，如果突然掉电、断网、屏幕卡死了，重新连接上去再编辑怎么办?

这时 Vi 可能会进入冲突模式，提示你如何处理。如果你处理后编辑内容没有作用，可以使用 ls -al 命令查看，一般被你所编辑的文件会产生一个临时文件，比如你编辑的文件叫 test，那么会有一个名为 test.swp 的临时文件，可以把它删除掉。

5.9 基础 Shell 编程

5.9.1 何为 Shell 编程

Shell 是一个作为用户与 Linux 系统间接口的程序，它允许用户向操作系统输入需要执行的命令，与 Windows 的命令提示符类似。Shell 就是位于内核与操作者之间的一层使用者界面，负责接收使用者输入的指令，然后将指令解释成内核能够了解的方式，传给内核去执行，再将结果传回到预设的输出终端。

Shell 即我们 Linux 下所使用的命令行，Shell 除了可以直接执行命令外，还可以由 Shell 的主程序来执行一些脚本语言。如果说 Linux 必须有的东西是什么，那么估计就是

Shell 和可执行程序了。绝大多数驱动程序和大量的应用软件最终都是使用 Shell 脚本进行处理。

Shell 语言在不同的解析器、不同的 Linux，甚至在 CentOS 和 OpenWrt 当中都有一定的区别，本节主要介绍 Shell 编程的基本方法。

5.9.2 Shell 的运行方式

- Interactive 交互式：当我们在系统中执行一个命令，并且根据屏幕反馈再考虑执行下一个命令时，这个就是 Shell 的交互式运行方式。每次交互只执行一条命令。
- Batch 批处理：使用者事先写好一个 Shell 程序，然后通过命令方式执行这个程序，这个程序中又包含很多其他的功能在一起批量实现，叫作批处理。系统启动时大量处理脚本都采用这种方式运行。

Shell 的代码与普通编程语言类似，具备变量和流程控制，但 Shell 是解析执行的，无需编译。

5.9.3 Hello World!

我们写的第一个程序通常是 Hello World! 这里也从 Hello World! 开始，用 Shell 来写。编写如下代码。


```
#!/bin/sh
echo "Hello World!"
```

- 第一句“#!/bin/sh”表示系统解析器的位置，一般是这个位置。
- 第二句是 echo 语法，显示出相关字符。

执行效果如下。

```
[root@localhost ~]# chmod +x hello
[root@localhost ~]# ./hello
Hello World!
```

5.9.4 变量

定义变量

- 变量名字 = 值，如 test_var="hello"。
- 首个字符必须为字母（a~z，A~Z）。
- 中间不能有空格，可以使用下划线（_）。
- 不能使用标点符号。
- 不能使用 Shell 里的关键字（可用 help 命令查看保留关键字）。

```
#!/bin/sh
test_var="Hello"
test_var2=100
echo "Hello World!"
~
~
~
~
~
~
```

使用变量

在变量名前加上 $ 符号即可使用这个变量，加入 {} 符号就可以让解析器识别字符边界。

```
#!/bin/sh
test_var="Hello"
test_var2=100
echo "${test_var} World ${test_var2}!"
~
~
~
~
~
~
```

执行效果如下。

```
[root@localhost ~]# ./hello
Hello World 100!
```

变量作用域

- **局部变量：**只在当前脚本或当前命令行环境中有效的变量。一般在脚本中声明的变量都属于局部变量。
- **环境变量：**所有程序和软件被通过命令行启动时可以访问到的变量，例如临时文件夹位置等。
- Shell **特殊变量：**一组特殊变量，用来帮助解决编写脚本时输入 / 输出的问题。

Shell 特殊变量

查看当前进程 ID，编写如下代码。

```
#!/bin/sh

echo $$
```

执行效果如下。

```
[root@localhost ~]# chmod +x test2
[root@localhost ~]# ./test2
1333
```

表 5-1　Shell 特殊变量

变量	含义
$0	当前脚本名
$[N]	通过命令行传递进入的参数，比如 $1、$2
$#	通过命令行传递进入的参数数量
$*	通过命令行传递进入的所有参数
$@	通过命令行传递进入的所有参数，与 $* 略有不同
$?	上个命令的退出状态或返回值
$$	当前 Shell 的进程 ID

Shell 特殊变量如表 5-1 所示，举例编写如下代码。

```
#!/bin/bash
echo "File Name: $0"
echo "First Parameter : $1"
echo "First Parameter : $2"
echo "Quoted Values: $@"
echo "Quoted Values: $*"
echo "Total Number of Parameters : $#"
```

执行效果如下。

```
[root@localhost ~]# chmod +x test3
[root@localhost ~]# ./test3
File Name: ./test3
First Parameter :
First Parameter :
Quoted Values:
Quoted Values:
Total Number of Parameters : 0
```

带参数执行效果如下。

```
[root@localhost ~]# ./test3 hello world
File Name: ./test3
First Parameter : hello
First Parameter : world
Quoted Values: hello world
Quoted Values: hello world
Total Number of Parameters : 2
```

命令替换

有时，我们需要将某些命令执行的结果存储到一个变量中，以便后面使用，这时就需要使用反引号 `` 来实现了，反引号就是主键盘数字 1 前面的那个按键。

举例，编写如下代码。

```
#!/bin/sh
now=`date`
echo "date now is ${now}"
```

执行结果如下。

```
date now is Tue Jan 20 16:38:45 CST 2015
```

5.9.5 运算符

算术运算符

- 加法运算符：+
- 减法运算符：−
- 乘法运算符：*
- 除法运算符：/
- 取余运算符：%
- 赋值运算符：=
- 相等比较符：==
- 不等比较符：!=

举例，编写如下代码。

```
#!/bin/sh
```

```
a=10
b=20
val=`expr $a + $b`
echo "a + b : $val"

val=`expr $a - $b`
echo "a - b : $val"

val=`expr $a \* $b`
echo "a * b : $val"

val=`expr $b / $a`
echo "b / a : $val"

val=`expr $b % $a`
echo "b % a : $val"

if [ $a == $b ]
then
   echo "a is equal to b"
fi

if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```


执行结果如下。

```
a + b : 30
a - b : -10
a * b : 200
b / a : 2
b % a : 0
a is not equal to b
```

关系运算符

关系运算符只支持数字，不支持字符串，除非字符串的结果是数字（比如真或假），关系运算符返回的值总是真或假。

- 相等运算符：-eq
- 不等于算符：-ne
- 左边是否大于右边：-gt
- 左边是否小于右边：-lt
- 左边是否大于等于右边：-ge

- 左边是否小于等于右边：-le

  举例，编写如下代码。

```
#!/bin/sh

a=10
b=20
if [ $a -eq $b ]
then
   echo "$a -eq $b : a is equal to b"
else
   echo "$a -eq $b: a is not equal to b"
fi

if [ $a -ne $b ]
then
   echo "$a -ne $b: a is not equal to b"
else
   echo "$a -ne $b : a is equal to b"
fi

if [ $a -gt $b ]
then
   echo "$a -gt $b: a is greater than b"
else
   echo "$a -gt $b: a is not greater than b"
fi

if [ $a -lt $b ]
then
   echo "$a -lt $b: a is less than b"
else
   echo "$a -lt $b: a is not less than b"
fi

if [ $a -ge $b ]
then
   echo "$a -ge $b: a is greater or  equal to b"
else
   echo "$a -ge $b: a is not greater or equal to b"
fi

if [ $a -le $b ]
then
   echo "$a -le $b: a is less or  equal to b"
else
   echo "$a -le $b: a is not less or equal to b"
fi
```

执行结果如下。

```
10 -eq 20: a is not equal to b
10 -ne 20: a is not equal to b
10 -gt 20: a is not greater than b
10 -lt 20: a is less than b
10 -ge 20: a is not greater or equal to b
10 -le 20: a is less or  equal to b
```

布尔运算符

- 非运算符：!
- 或运算符：-o
- 与运算符：-a

布尔运算符在 Shell 中使用得比较少，这里只例举非运算符。

```
#!/bin/sh

a=10
b=20

if [ $a != $b ]
then
   echo "$a != $b : a is not equal to b"
else
   echo "$a != $b: a is equal to b"
fi
```

执行结果如下。

```
10 != 20 : a is not equal to b
```

字符串运算符

- 字符串相等符：=
- 字符串不等符：!=
- 字符串长度为 0：-z
- 字符串长度不为 0：-n
- 字符串为空：str

举例，编写如下代码。

```
#!/bin/sh
```

```
a="abc"
b="efg"

if [ $a = $b ]
then
   echo "$a = $b : a is equal to b"
else
   echo "$a = $b: a is not equal to b"
fi

if [ $a != $b ]
then
   echo "$a != $b : a is not equal to b"
else
   echo "$a != $b: a is equal to b"
fi

if [ -z $a ]
then
   echo "-z $a : string length is zero"
else
   echo "-z $a : string length is not zero"
fi

if [ -n $a ]
then
   echo "-n $a : string length is not zero"
else
   echo "-n $a : string length is zero"
fi

if [ $a ]
then
   echo "$a : string is not empty"
else
   echo "$a : string is empty"
fi
```

执行结果如下。

```
abc = efg: a is not equal to b
abc != efg : a is not equal to b
-z abc : string length is not zero
-n abc : string length is not zero
abc : string is not empty
```

文件测试运算符

这些是 Shell 中最常用的指令了，经常要判断文件的各种情况。

- 检测文件是否为块设备：-b file
- 检测文件是否为字符设备：-c file
- 检测文件是否为目录：-d file
- 检测文件是否为普通文件：-f file
- 检测文件是否为管道文件：-p file
- 检测文件是否可读：-r file
- 检测文件是否可写：-w file
- 检测文件是否可执行：-x file
- 检测文件是否为空：-s file
- 检测文件是否存在：-e file

举例，编写如下代码。

```
#!/bin/sh

file="/var/www/tutorialspoint/unix/test.sh"

if [ -r $file ]
then
   echo "File has read access"
else
   echo "File does not have read access"
fi

if [ -w $file ]
then
   echo "File has write permission"
else
   echo "File does not have write permission"
fi

if [ -x $file ]
then
   echo "File has execute permission"
else
   echo "File does not have execute permission"
fi

if [ -f $file ]
then
   echo "File is an ordinary file"
else
   echo "This is sepcial file"
fi

if [ -d $file ]
```

```
then
   echo "File is a directory"
else
   echo "This is not a directory"
fi

if [ -s $file ]
then
   echo "File size is not zero"
else
   echo "File size is zero"
fi

if [ -e $file ]
then
   echo "File exists"
else
   echo "File does not exist"
fi
```

执行结果

```
File has read access
File has write permission
File does not have execute permission
File is an ordinary file
This is not a directory
File size is not zero
File exists
```

5.9.6 注释语法

注释表示所写内容不被解析器执行，仅仅用来说明。Shell 不支持多行注释语法，每次只能注释一行，以 # 开头的内容到换行位置表示注释。

举例，编写如下代码。

```
#!/bin/sh
# echo "Hello"
echo "World!"
```

执行结果如下。

```
World!
```

5.9.7 字符串

单字符串

```
abc='only string'
```

单字符串不支持任何变量，所有信息都直接输出。

转义字符串

```
ddd='hello'
abc="${ddd} World is \"Smart Router\"!"
```

在转义字符串中可以直接显示变量，也支持转义符号。

字符串拼接

将两个字符串合并在一起。

```
your_name="axwrt"
greeting=$your_name" is branch of openwrt !"
echo $greeting
```


获得字符串长度

```
string="abcd"
echo ${#string}
```

提取字符

```
string="Hello World!"
echo ${string:1:4} # 输出 Helo
```

5.9.8 条件判断

单结果判断 if

语法格式：

```
if [ 表达式 ]
then
   如果表达式结果成立，则执行……
```

```
fi
```

举例，编写如下代码。

```
#!/bin/sh
a=10
b=20
if [ $a == $b ]
then
   echo "a is equal to b"
fi
```

双结果判断 if else

```
if [ 表达式 ]
then
   如果表达式结果成立，则执行……
else
   如果表达式结果不成立，则执行……
fi
```

举例，编写如下代码。

```
#!/bin/sh

a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
else
   echo "a is not equal to b"
fi
```

多结果判断 if elif fi

```
if [ 表达式 1 ]
then
   如果表达式 1 结果成立，则执行……
elif [ 表达式 2 ]
then
   如果表达式 2 结果成立，则执行……
elif [ 表达式 3 ]
```

```
then
   如果表达式 3 结果成立，则执行……
else
   如果表达式全部结果都不成立，则执行……
fi
```

举例，编写如下代码。

```
#!/bin/sh

a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```


5.9.9 循环表达

循环 for

```
for 变量 in 列表
do
    指令 1
    指令 2
    ……
    更多指令……
done
```

举例，根据数字循环。

```
#!/bin/sh
for loop in 10 20 21 22 25
do
    echo "The value is: $loop"
done
```

执行结果如下。

```
The value is: 10
The value is: 20
The value is: 21
The value is: 22
The value is: 25
```

循环 while

```
while 指令内容
do
   在指令内容为真的情况下执行
done
```

举例，编写如下代码。

```
#!/bin/sh
i=0
while [ $i -lt 5 ]
do
    i=`expr $i + 1`
    echo ${i}
done
```

执行结果如下。

```
1
2
3
4
5
```

中断全部循环 break/continue

break 与 continue 的用法完全相同，唯一的区别就是 break 退出全部循环（如果有多级循环），而 continue 只退出一级。

举例，编写如下代码。

```
#!/bin/sh
i=0
while [ $i -lt 5 ]
do
    i=`expr $i + 1`
```

```
    echo ${i}
    if [ $i -eq 3 ]; then
        echo "count 3 break now"
        break
    fi
done
```

执行结果如下。

```
1
2
3
count 3 break now
```

5.9.10 输入 / 输出重定向

“有时候，有时候，我会相信一切有尽头。”事实上，真的有时候，我们不希望 Shell 代码的执行结果输出到屏幕上。

我们在屏幕上能看到的信息虽然是在同一个地方显示的，其实来源是两个不同的途径，也叫作管道，它们分别是：

- STDOUT（标准输出），以字符 1 为代表，大部分的打印或显示函数以及语法都是通过这个管道输出的；
- STDERR（标准错误输出），以字符 2 为代表，大部分的错误，只要程序写得规范，错误信息就是通过这个管道输出的。

也就是说，不可能让信息不输出，但是可以让它们不显示在屏幕上，而显示在别的地方，比如显示一个不存在的地方或一个文件中。

举例，使用右尖括号 > 将 STDOUT 改变到 display 文件中。

```
[root@localhost ~]# ls > display
[root@localhost ~]# cat display
anaconda-ks.cfg
count
count2
display
hello
install.log.syslog
t1
test
test2
test3
test4
```

举例，使用两个右尖括号 >> 追加 STDOUT 信息到 display 文件中，而不是覆盖掉已有的。

```
[root@localhost ~]# ls >> display
[root@localhost ~]# cat display
anaconda-ks.cfg
count
count2
display
hello
install.log.syslog
t1
test
test2
test3
test4
anaconda-ks.cfg
count
count2
display
hello
install.log.syslog
t1
test
test2
test3
test4
```

5.9.11 使用黑洞，一劳永逸

有时我们写一个程序，负责启动一个服务，但是那个服务经常乱显示，也不适合将显示的信息放在一个文件中，显示的东西实在没有任何用处，那就得靠 Linux 来帮忙啦！ Linux 有一个很有意思的设备文件，叫作黑洞，这个文件的特点是：它永远是空的，凡是扔到它里面的东西都将一去不复返。它就是 /dev/null。

举例，没有扔 STDERR 的时候，错误信息都显示在屏幕上了。

```
[root@localhost ~]# ls device > /dev/null
ls: cannot access device: No such file or directory
```

举例，将 STDERR 扔到黑洞中。

```
[root@localhost ~]# ls device 2> /dev/null
```

举例，将 STDOUT、STDERR 全部扔到黑洞中。

```
[root@localhost ~]# ls device >/dev/null 2>&1
```

Shell 编程是非常博大精深的，要想真的全面地学习 Shell 编程，完全可以写成一本厚书。若是读者在测试过程中发现有问题，多半是语法格式不对，Shell 对语法格式的要求很严格。希望本章内容可以起到参考作用，为后面我们所要学习的知识打好基础。

第6章 OpenWrt 刷机

“道冲，而用之或不盈。渊兮，似万物之宗。挫其锐，解其纷，和其光，同其尘。湛兮似或存。吾不知谁之子，象帝之先。”

从本章开始到后面的章节主要介绍 OpenWrt 的各种玩法，包括智能路由器的基本网络配置、存储器扩展、智能路由媒体中心、连接音箱播放音乐、实现建站、USB 摄像头和智能路由外设计应用实例。

本章主要介绍两种 OpenWrt 刷机方法，一种是 U-Boot 的 TFTP 刷机，另一种是 U-Boot 的 Web 刷机。

6.1 通过串口进入系统

在 MR808 开发板上电之前必须要注意：核心板一定要插入底板，然后接上串口调试线，再上电。OpenWrt 在编译时已经默认打开串口作为本地终端了，通过串口终端，使用者可以像操作自己的计算机一样来操作开发板，可以实现 SSH 或 Telnet 做不了的事情，比如查看内核的一些信息，控制引导系统刷机等。

6.1.1 操作板子注意事项

- 注意要释放掉身上的静电，摸一下其他接地的物品，甚至摸一下墙都会有效果。
- 在连接好需要的调试线路之前不能上电。
- 注意板子不要出现短路情况，不要将其他的东西放在板子的上面或下面，以免形成短路。

6.1.2 连接串口步骤

1 在计算机上安装 USB 转串口线驱动程序。

驱动程序下载地址：http://pan.baidu.com/s/1hspbap2

2 将 USB 转串口线接入计算机的 USB 接口，等待驱动处理完成，然后查看 PC 设备管理器，确认 COM 口的编号，因为 USB 接入计算机位置不同，会导致 COM 口不一样，USB 转串口驱动程序安装成功如图 6-1 所示。

图 6-1　USB 转串口驱动程序安装成功

3 打开 PuTTY 软件，Connection Type 选择 Serial。在 Serial Line 中填写 COM 口编号，在 Speed（速率）中填写 115200，填写完毕后单击“Open”，如果没报错，直接出现黑窗口表示串口打开成功。PuTTY 配置界面如图 6-2 所示。

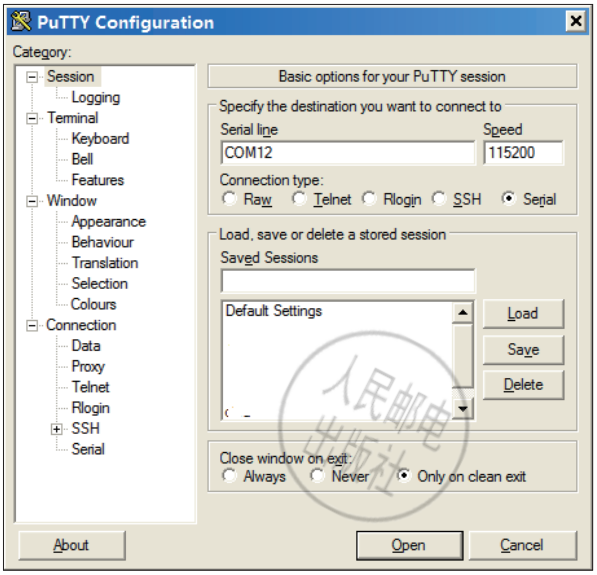


图 6-2　PuTTY 配置界面

4 将 USB 转串口线的另外一端接入 MR808 开发板底板的 TTL 接口上，然后将底板上电，并且按下开关，会看到如下信息。

```
U-Boot 1.1.3 - Modified by Manfeel (Jan 20 2015 - 23:02:47)

Board: Ralink APSoC


 __  __                ______          _
|  \/  |              |  ____|        | |
| \  / | __ _ _ __ | |__ ___   ___| |
| |\/| |/ _` | '_ \|  __/ _ \/ _ \ |
| |  | | (_| | | | | | |  __/  __/ |
|_|  |_|\__,_|_| |_|_|  \___|\___|_|

DRAM:  64 MB
relocate_code Pointer at: 83fa8000
```

```
enable ephy clock...done. rf reg 29 = 5
SSC disabled.
spi_wait_nsec: 28
spi device id: ef 40 18 0 0 (40180000)
find flash: W25Q128FV
raspi_read: from:30000 len:1000
*** Warning - bad CRC, using default environment
```

5 看到的信息会大量滚动，最后进入系统，完成连接这一步骤。

USB 转串口线看起来简单，却可能会出现以下问题。

- PuTTY 打不开 COM 设备：驱动有问题、参数填写有问题或 TTL 线有问题。
- 开发板上电以后屏幕什么都不显示：连接线有问题。
- 开发板烧了，赶紧联系硬件支持。
- 屏幕有显示，但显示的是乱码：可能是 Speed（速率）没设置正确，也可能是线路接触不良。重新连接所有设备，包括重插串口板和重新给开发板上电。
- 所有信息都能正常显示，但是按回车键没反应，也无法输入信息：可能是线有问题，或者是没到能输入的时候。

6.2 认识 U-Boot

经过上面的步骤，可能细心的人会发现我们已经进入系统了，可以输入一些标准的 Linux 命令。那么不要着急，我们再回来讲一下 BootLoader 引导系统。

BootLoader 在操作系统内核运行之前运行，可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，以便为最终调用操作系统内核准备好合适的环境。在嵌入式系统中，通常并没有像 BIOS 那样的固件程序，因此整个系统的加载启动任务就完全由 BootLoader 来完成了。

在我们的开发板上，操作系统启动前，BootLoader 要对一些基本的电子设备进行检测，同时完成 CPU 所要求的初始化，再启动 Flash 上的 Linux 内核。

MT7620 这款 CPU 所使用的引导程序名字叫 U-Boot，U-Boot 是德国 DENX 小组开发的用于嵌入式 CPU 的引导程序（U-Boot 也支持 x86 架构），它支持 10 来种不同的操作系统。它小巧、速度快，已经成为了 Atheros 和 Ralink 产品的标配引导系统，并且在所有嵌入式的引导系统中市场占有率最大。

很多书籍都详细讲解了 U-Boot 是如何引导操作系统的，其实 U-Boot 如何引导都是 CPU 厂家提供的 SDK 里要做的事，所以本书改变一下说法，结合所使用的开发板来讲讲系统到底是如何启动的。基本流程不外乎：开发板上电→晶体振荡器发起频率振动→ CPU 启动内部代码→ U-Boot 启动→ U-Boot 根据参数设置倍频→ U-Boot 进行一些其他的内部寄存器配置→ U-Boot 启动操作系统。

首先开发板上电，上电后晶体振荡器开始发起一个指定频率的振动，这时 CPU 就根据晶体振荡器的频率进行了启动，CPU 工作在晶体振荡器频率下启动自己的内部代码（CPU 内部有一小段代码和一点点内存，用来做最基础的事情，并且查找 BootLoader 在哪），然后启动 U-Boot。U-Boot 将 Flash 中的内核加载到内存，之后将控制权交给操作系统内核，操作系统运行。

所谓刷机就是刷写新的系统到板载闪存上，使用 U-Boot 刷写是很安全的，即使刷写坏了也可以重新刷写，一直到好用为止。第一次刷机要从 U-Boot 操作开始，在开发板上提供了两个 U-Boot 以及两种刷机方法，一种是 U-Boot 的 TFTP 刷机，另一种是 U-Boot 的 Web 刷机。下面详细介绍两种方法。

6.3 U-Boot 的 TFTP 刷机

很多人会宣传自己的 U-Boot 支持刷不死，其实没有那回事，用 U-Boot 不管怎么刷机，都不可能把 U-Boot 刷死，除非你在刷 U-Boot。只能说刷写过程中出现异常，导致刷上去的系统有问题，只要重刷就行了。OpenWrt 固件下载地址如下。

Tips

百度网盘下载：http://pan.baidu.com/s/1qYTwyHe

服务器下载地址：http://www.maker1000.com/makerouter/system/openwrt-ramips-mt7620n-mr808-squashfs-sysupgrade.bin

6.3.1 TFTP 的准备

（1）将开发板用网线与计算机连接起来，并且将计算机的 IP 设置成为 192.168.128.10，不设置网关和 DNS 服务器。

（2）下载安装 TFTP 服务端软件，并且安装到计算机上，然后启动该程序，注意计算机的防火墙要打开允许，否则开发板访问不到计算机。

Tips

TFTP32 服务端程序百度网盘下载地址：http://pan.baidu.com/s/1slVwinV

TFTP32 服务端程序服务器下载址：http://www.maker1000.com/makerouter/software/TFTP32.zip

（3）单击 TFTP 主界面的 settings（设置）。在 Start Services 中选择打开 TFTP Server。在 TFTP 的标签中 Base Directory 处选 TFTP 所在文件夹，表示要刷机的文件在 TFTP 的文件夹下面。其他按默认即可，单击“确认”。

（4）在 TFTP 主界面 Server Interface 处选择 192.168.128.10 那个 IP 作为 TFTP 的服务位置。

（5）将固件复制到 TFTP 文件夹下面。

6.3.2 TFTP 刷机流程

（1）给 MR808 开发板重新上电，待看到屏幕显示选择菜单的瞬间按下键盘上的数字 2，选择“Load system code then write to Flash via TFTP.”模式，PuTTY 显示界面如下所示。

```
Please choose the operation:
   1: Load system code to SDRAM via TFTP.
   2: Load system code then write to Flash via TFTP.
   3: Boot system code via Flash (default).
   4: Entr boot command line interface.
   7: Load Boot Loader code then write to Flash via Serial.
   9: Load Boot Loader code then write to Flash via TFTP.
```

（2）系统将会提示是否确认进行这个操作，用键盘输入 Y 表示确认执行。PuTTY 显示界面如下所示。

```
2: System Load Linux Kernel then write to Flash via TFTP.
 Warning!! Erase Linux in Flash then burn new one. Are you sure?(Y/N)
```

（3）在 Input device IP 位置填写上 MR808 开发板的 IP 地址，填写 192.168.128.1 即可。在 Input server IP 位置填写你的计算机的 IP 地址 192.168.128.10 即可。在 Input Linux Kernel filename 位置填写固件名称 openwrt-ramips-mt7620n-mr808-squashfs-sysupgrade.bin。PuTTY 显示界面如下。

```
Please Input new ones /or Ctrl-C to discard
        Input device IP (192.168.1.1) ==:192.168.128.1
        Input server IP (192.168.1.100) ==:192.168.128.10
        Input Linux Kernel filename () ==:openwrt-ramips-mt7620n-mr808-
squashfs-sysupgrade.bin
```

（4）如果填写正确，即可进入刷机流程，U-Boot 先下载固件，再烧写，完成后会重启，然后屏幕就“噼里啪啦”跳出来一堆东西，表示正在进入操作系统。启动后 5 秒之内将完成可写分区格式化，最终就完成了全部流程。在烧写过程中，PuTTY 显示界面如下。

```
NetTxPacket = 0x83FE2A40

 KSEG1ADDR(NetTxPacket) = 0xA3FE2A40

 NetLoop,call eth_halt !

 NetLoop,call eth_init !
```

```
Trying Eth0 (10/100-M)

 Waitting for RX_DMA_BUSY status Start... done


 ETH_STATE_ACTIVE!!
TFTP from server 192.168.128.10; our IP address is 192.168.128.1
Filename 'openwrt-ramips-mt7620n-mr808-squashfs-sysupgrade.bin'.

 TIMEOUT_COUNT=10,Load address: 0x80100000
Loading: Got ARP REPLY, set server/gtwy eth addr (00:26:6c:10:93:0e)
Got it
#################################################################
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        #################################################################
        ##############################################
done
Bytes transferred = 2883588 (2c0004 hex)
NetBootFileXferSize= 002c0004
raspi_erase_write: offs:50000, count:2c0004
raspi_erase: offs:50000 len:2c0000
.......................................
raspi_write: to:50000 len:2c0000
.......................................
raspi_read: from:50000 len:10000
raspi_read: from:60000 len:10000
raspi_read: from:70000 len:10000
......
raspi_read: from:2d0000 len:10000
raspi_read: from:2e0000 len:10000
raspi_read: from:2f0000 len:10000
raspi_read: from:300000 len:10000
raspi_read: from:310000 len:10000
raspi_erase: offs:310000 len:10000
.
raspi_write: to:310000 len:10000
.
raspi_read: from:310000 len:10000
Done!
## Booting image at bc050000 ...
raspi_read: from:50000 len:40
   Image Name:   MIPS OpenWrt Linux-3.10.49
   Image Type:   MIPS Linux Kernel Image (lzma compressed)
   Data Size:    1028808 Bytes = 1004.7 kB
```

```
   Load Address: 80000000
   Entry Point:  80000000
raspi_read: from:50040 len:fb2c8
   Verifying Checksum ... OK
   Uncompressing Kernel Image ...
hdr->ih_load = 0x00000080
data pointer = 0x80c00000
first 8 bytes data = 8000006D,2EEC3400
data length = 1028808
OK
No initrd
## Transferring control to Linux (at address 80000000) ...
manfeel note, bootargs is : <NULL>
## Giving linux memsize in MB, 64
83f477b0:linux_env[0] = memsize=64
83f477bb:linux_env[1] = initrd_start=0x00000000
83f477d3:linux_env[2] = initrd_size=0x0
83f477e3:linux_env[3] = flash_start=0x00000000
83f477fa:linux_env[4] = flash_size=0x1000000

Starting kernel ...
......
[   77.170000] jffs2: notice: (760) jffs2_build_xattr_subsystem:
complete building xattr subsystem, 0 of xdatum (0 unchecked, 0 orphan)
and 0 of xref (0 dead, 0 orphan) found.
procd: - init complete -
```

6.3.3 可能遇到的问题

- 填写或选项过程中有错误：重新给 MR808 开发板上电开始。
- 都填写完成但是无法完成刷机：可能是网络没通，比如 TFTP 被防火墙禁止了或网线没插好。
- 刷机一半失败：请重新操作。

6.4 U-Boot 的 Web 刷机

如果板子是支持 Web 刷机的，U-Boot 即可完成这个操作流程，接下来介绍如何进行 Web 刷机。

6.4.1 触发按钮准备

（1）将按钮板接入底板上名为 I2C_SCLK/IO=2 的一排针脚，注意针脚方向不要接错。内侧为 IO，中间为 3.3V，外侧为 GND。

（2）将 MR808 开发板用网线与计算机连接起来，并且将计算机的 IP 设置成为 192.168.1.10，不设置网关和 DNS 服务器。注意防止 IP 地址冲突，不要出现计算机还在用其

他网卡上网，同时也用了 192.168.1.X 这个网段的 IP 地址的情况。

（3）按下按钮板的按钮不动，然后给 MR808 开发板上电。等待屏幕显示如下信息后松开按钮板的按钮，表示 MR808 开发板已经进入 U-Boot 的 Web 刷机准备状态。

```
WPS(GPIO2) button pressed!
raspi_read: from:40028 len:6


 NetTxPacket = 0x83FE2A40

 KSEG1ADDR(NetTxPacket) = 0xA3FE2A40
Trying Eth0 (10/100-M)

 Waitting for RX_DMA_BUSY status Start... done


 ETH_STATE_ACTIVE!!
HTTP server is starting at IP : 192.168.1.1
HTTP server is ready!

Keep going WEB failsafe, rx_counter=0 , GoGoGo!!!
```

（4）屏幕中的信息显示 MR808 开发板目前的 IP 为 192.168.1.1。

（5）打开浏览器访问这个 IP 地址，将会显示出刷机界面。

（6）单击“选择文件”选择上刚才已经下载的教学固件文件。单击“上传”开始刷机流程。

（7）单击“上传”后立即转换到 PuTTY 的界面，可以观察到上传的流程和刷机流程，显示信息如下所示。

```
Request for: /
hs->upload=1355,uip_mss=1460
Request for: /style.css
hs->upload=1310,uip_mss=1460
Request for: /js/lang.js
hs->upload=1324,uip_mss=1460
Request for: /img/logo.png
hs->upload=1119,uip_mss=1460
Request for: /favicon.ico
## /favicon.ico not found in shared directory!
hs->upload=604,uip_mss=1460
Data will be downloaded at 0x80100000 in RAM
Upgrade type: firmware
Upload file size: 2883588 bytes
Loading: ########################################
         ########################################
```

```
         ########################################
         ########################################
....
         ########################################
         ########################################
         ############################

Request for: /style.css
hs->upload=1310,uip_mss=1460
HTTP upload is done! Upgrading...


****************************
*    FIRMWARE UPGRADING    *
* DO NOT POWER OFF DEVICE! *
****************************

raspi_erase_write: offs:50000, count:2c0004
raspi_erase: offs:50000 len:2c0000
.........................................
raspi_write: to:50000 len:2c0000
.........................................
raspi_read: from:50000 len:10000
raspi_read: from:60000 len:10000
......
raspi_read: from:300000 len:10000
raspi_read: from:310000 len:10000
raspi_erase: offs:310000 len:10000
.
raspi_write: to:310000 len:10000
.
raspi_read: from:310000 len:10000
Done!
HTTP ugrade is done! Rebooting...
......
[   77.170000] jffs2: notice: (760) jffs2_build_xattr_subsystem:
complete building xattr subsystem, 0 of xdatum (0 unchecked, 0 orphan)
and 0 of xref (0 dead, 0 orphan) found.
procd: - init complete -
```

6.4.2 可能遇到的问题

- 无法访问 192.168.1.1：可能计算机 IP 地址或网络部分有错，也可能是网线连接错误。
- 填写完成，但是 PuTTY 显示了错误的信息：可能上传失败，清空浏览器缓存，重新打开浏览器，重复操作整个流程。
- 刷机一半失败：请重新操作。

第7章 分区、外网与 OPKG

7.1 了解分区

想必很多人都知道分区的概念吧？磁盘分成不同的分区来工作，在路由器的 Flash 上也是有分区的。

上一章介绍的首次刷机完成后，再过一段时间会有以下提示。

```
[   77.170000] jffs2: notice: (760) jffs2_build_xattr_subsystem: complete
building xattr subsystem, 0 of xdatum (0 unchecked, 0 orphan) and 0 of
xref (0 dead, 0 orphan) found.
```

这段话的意思是，使用 jffs2 文件系统完成了格式化。好啦，我们先不管什么是 jffs2 了，先来说说分区吧。

在 Linux 系统中对闪存类存储器是采用 MTD（内存技术设备）类设备驱动实现的，MTD 是用于访问内存类设备（ROM、Flash）的 Linux 驱动子系统。它的主要目的是使 Flash 闪存类设备更加容易被访问，为此它在硬件和上层提供了一个抽象的接口使得在操作系统下我们可以像操作硬盘一样操作这类设备。仔细观察过 Linux 启动信息的朋友会看到以下一段话。

```
[    0.290000] 4 ofpart partitions found on MTD device spi32766.0
[    0.290000] Creating 4 MTD partitions on "spi32766.0":
[    0.300000] 0x000000000000-0x000000030000 : "u-boot"
[    0.310000] 0x000000030000-0x000000040000 : "u-boot-env"
[    0.310000] 0x000000040000-0x000000050000 : "factory"
[    0.320000] 0x000000050000-0x000001000000 : "firmware"
[    0.330000] 0x00000014b308-0x000001000000 : "rootfs"
[    0.330000] mtd: partition "rootfs" must either start or end on erase
block boundary or be smaller than an erase block -- forcing read-only
[    0.350000] mtd: device 4 (rootfs) set to be root filesystem
[    0.350000] mtd: partition "rootfs_data" created automatically,
ofs=0x310000, len=0xcf0000
[    0.360000] 0x000000310000-0x000001000000 : "rootfs_data"
```

这段话的意思是，系统在 SPI（SPI 是我们所使用的 Flash 接口标准，路由器一般都用它，价格便宜量又足！）设备上创建了 4 个 MTD 分区，这几个分区的说明如表 7-1 所示。

表 7-1 分区说明

分区 ID 号	分区位置	分区容量	分区作用
u-boot	0x000000000000~0x000000030000	192KB	引导程序
u-boot-env	0x000000030000~0x000000040000	64KB	引导程序配置
factory	0x000000040000~0x000000050000	64KB	MT7620 初始参数
firmware	0x000000050000~0x000001000000	15.68MB	固件分区
rootfs	0x00000014b308~0x000001000000	14.70MB	固件分区 文件系统子集
rootfs_data	0x000000310000~0x000001000000	12.93MB	固件分区 文件系统子集 可写分区子集

由于嵌入式的 Flash 容量都很小，没有调整的必要，所以分区都是固定的，也因此不需要“分区表”这种在计算机上才有的东西。在路由器的 Flash 中，分区的位置都是固化好的。

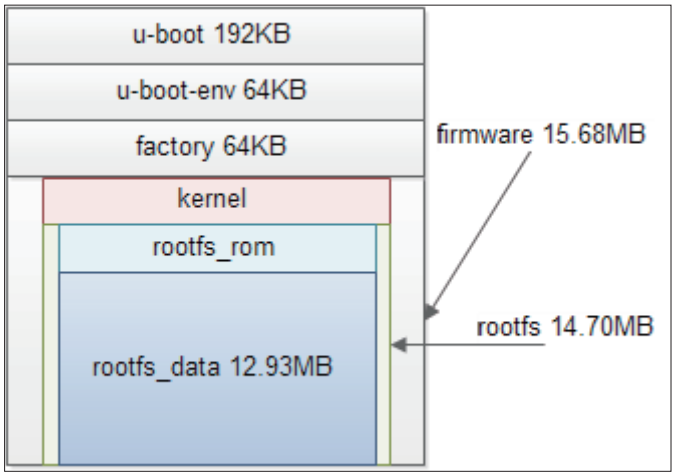


图 7-1 Flash 分区

MR808 开发板的 Flash 容量是 16MB，Flash 分区如图 7-1 所示。

（1）分区存在子分区，比如 kernel 和 rootfs 就是 firmware 的子分区。而 rootfs_rom、rootfs_data 就是 rootfs 的子分区。

（2）整个分区从 0x000000000000 开始，到 0x000001000000 结束，如果你觉得看起来不爽，可以用你的计算机中的计算器软件将十六进制转换成十进制。

（3）kernel 分区的容量计算公式如下，为什么这么写不再解释，请看前面那个表的分

区位置来琢磨。

```
( 0x00000014b308 - 0x000000050000 ) = 换算为十进制 / 1024 / 1024 = 0.98MB
```

（4）rootfs_rom 分区的容量计算公式如下，也不解释了。

```
(0x000000310000 - 0x00000014b308 ) = 换算为十进制 / 1024 / 1024 = 1.76MB
```

分区的容量不是一成不变的，随着固件的变化，其 firmware 部分的容量也会改变，不过没有关系，只要学会上面的公式就可以了解分区是怎么存在的。

7.1.1 查看系统 MTD 分配

```
root@OpenWrt:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00030000 00010000 "u-boot"
mtd1: 00010000 00010000 "u-boot-env"
mtd2: 00010000 00010000 "factory"
mtd3: 00fb0000 00010000 "firmware"
mtd4: 00eb4cf8 00010000 "rootfs"
mtd5: 00cf0000 00010000 "rootfs_data"
```

7.1.2 查看系统 MTD 的分区

```
root@OpenWrt:~# cat /proc/partitions
major minor  #blocks  name

  31        0        192 mtdblock0
  31        1         64 mtdblock1
  31        2         64 mtdblock2
  31        3      16064 mtdblock3
  31        4      15059 mtdblock4
  31        5      13248 mtdblock5
```

7.1.3 将非文件系统分区读出来

factory 分区是非文件系统的，里面保存的是配置，比如当前这台机器的 MAC 地址，可以通过一个十六进制读取命令直接对其读取。

如前面所讲，factory 分区的设备文件为 mtd2，就是 /dev/mtd2。

```
root@OpenWrt:~# hexdump -C /dev/mtd2
```

这部分内容不太容易理解，可以查看芯片厂提供的资料，芯片厂一般不会公开这部分

的参数，但是我们可以看看芯片厂的资料里哪个像 MAC 地址，这就是芯片出厂时的 MAC 地址。

7.2 文件系统

下面开始深入地了解文件系统与上面分区的关系，系统有 rootfs_rom 这个隐藏的只读文件系统，也有 rootfs_data 这个可写的文件分区，它们之间是什么关系呢？这是 OpenWrt 设计的一个有意思的地方。

（1）首先，引导程序启动内核完成之后，由内核加载 roofs_rom 只读分区部分来完成系统的初步启动。

（2）rootfs_rom 只读分区采用的是 Linux 内核支持的 squashFS 文件系统（一种压缩只读文件系统），加载完毕后将其挂载到 /rom 目录（同时也挂载为 / 根目录）。

（3）系统将使用 JFFS2 文件系统格式化的 rootfs_data 可写文件分区并且将这部分挂载到 /overlay 目录。

（4）系统再将 /overlay 透明挂载为 / 根目录。

（5）最后将一部分内存挂载为 /tmp 目录。

（6）挂载情况如下。

```
root@OpenWrt:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
rootfs                   13248       464     12784   4% /
/dev/root                 1792      1792         0 100% /rom
tmpfs                    30940     15116     15824  49% /tmp
/dev/mtdblock5           13248       464     12784   4% /overlay
overlayfs:/overlay       13248       464     12784   4% /
tmpfs                      512         0       512   0% /dev
```

7.2.1 透明挂载 / 根目录

OpenWrt 设计的一个特点是：系统先将 rootfs_rom 挂载为 / 根目录，这样就具备了一个完整的系统，然后再将 rootfs_data 以透明方式挂载在 / 根目录上。OpenWrt 透明挂载处理流程如图 7-2 所示，这样重叠之后的效果是：

（1）我们所看到的根文件系统是由 rootfs_rom 和 rootfs_data 两个分区组合在一起的；

（2）当我们修改一个任何位置的文件的时候，所做的修改在 rootfs_data 里都会记录；

（3）当我们删除一个文件的时候，所做的修改在 rootfs_data 里都会记录；

（4）当我们增加一个文件的时候，所做的修改在 rootfs_data 里都会记录；

（5）当我们读取文件的时候，首先检测 rootfs_data 里的状态，再检测 rootfs_rom 里的内容，一直到最后给你一个结果。

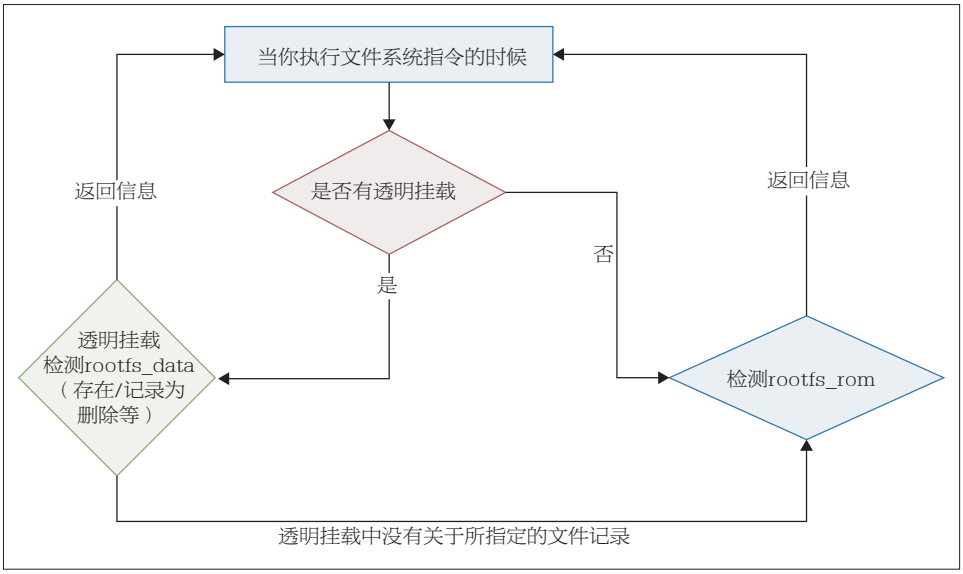


图 7-2　OpenWrt 透明挂载处理流程示意图

这样做的好处和坏处为：

（1）当对文件进行操作的时候，比如我们修改了一个名字为 abc 的文件，那么同时在 /rom 里还有修改之前的那个 abc，同时在 /overlay 里有修改之后的 abc，所占的空间是倍增的；

（2）系统不论任何时候，只要通过简单地删除掉 /overlay 里所有文件，就能达到复原的效果。

一个简单的例子如下。

```
root@OpenWrt:~# ls /31test
ls: /31test: No such file or directory
root@OpenWrt:~# ls /overlay/31test
ls: /overlay/31test: No such file or directory
root@OpenWrt:~# touch /31test
root@OpenWrt:~# ls /31test
/31test
root@OpenWrt:~# ls /overlay/31test
/overlay/31test
root@OpenWrt:~# rm /31test
root@OpenWrt:~# ls /overlay/31test
ls: /overlay/31test: No such file or directory
```

不过不建议直接在 /overlay 里删除，否则会出现不同步的情况！

7.2.2　基本系统演示

OpenWrt 中的系统指令没有使用 x86 下的指令包，因为那些指令包容量太大，它

使用了 BusyBox 项目。BusyBox 是一个集成了一百多种常用 Linux 命令和工具的软件（OpenWrt 下编译可以选择需要哪些指令）。除了包含常规指令之外，BusyBox 还包含了 grep find mount 以及 telnet http 等复杂工具。它被戏称为“Linux 下的瑞士军刀”。看看有多少个命令是 BusyBox 实现的命令如下所示。

```
root@OpenWrt:~# ls -l /bin/
```

常用文件夹：

（1）/etc/：存放着系统全部的配置文件。

（2）/etc/init.d/：存放着启动的服务脚本。

（3）/etc/config/：存放着 OpenWrt 的配置文件，包括网络等。

（4）/tmp/：除临时文件之外还存放着动态的配置文件。

（5）/tmp/TZ：系统启动后所使用的时区参数。

（6）其他还有很多很多，我们以后慢慢来了解。

7.2.3 第一次接入网络

从后续的章节开始，我们要确保系统经常保持联网的状态。下面介绍常见连接入网络的方法，使用一条可以连接外网的网线接入设备的 WAN 口（在 MR808 开发板底板靠近 USB 一侧的网口），接入成功后，系统会有如下反应。

```
root@OpenWrt:~# [ 6815.200000] ralink_soc_eth 10100000.ethernet eth0:
port 0 link up (100Mbps/Full duplex)
[ 6815.210000] br-lan: port 1(eth0.1) entered forwarding state
[ 6815.220000] br-lan: port 1(eth0.1) entered forwarding state
[ 6817.220000] br-lan: port 1(eth0.1) entered forwarding state
```

以上内容表示网络连接成功。这时可以使用命令 ifconfig 查看设备是否获得了上游路由器 DHCP 分配下来的 IP 地址（系统在默认情况下，WAN 口动态获取 IP 地址）。

```
root@OpenWrt:~# ifconfig
br-lan    Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fdb4:329c:dc0b::1/60 Scope:Global
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:372 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:34464 (33.6 KiB)

eth0      Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
```

```
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:431 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2478 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:131949 (128.8 KiB)  TX bytes:816662 (797.5 KiB)
          Interrupt:5

eth0.1    Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:140 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:12232 (11.9 KiB)

eth0.2    Link encap:Ethernet  HWaddr 64:51:7E:33:98:45
          inet addr:192.168.0.175  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::6651:7eff:fe33:9845/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:416 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2333 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:121890 (119.0 KiB)  TX bytes:792463 (773.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:55313 errors:0 dropped:0 overruns:0 frame:0
          TX packets:55313 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3764208 (3.5 MiB)  TX bytes:3764208 (3.5 MiB)
```

网络设备 eth0.2 为 WAN 口设备，我们的板子已经获得了 IP 地址了。使用 ping 命令测试网络是否通（可用组合键 Ctrl+C 中断测试）。

```
root@OpenWrt:~# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: seq=0 ttl=64 time=0.506 ms
64 bytes from 192.168.0.1: seq=1 ttl=64 time=0.405 ms
^C
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.405/0.455/0.506 ms
root@OpenWrt:~# ping www.geeqee.com
PING www.geeqee.com (114.215.133.69): 56 data bytes
64 bytes from 114.215.133.69: seq=0 ttl=53 time=33.405 ms
64 bytes from 114.215.133.69: seq=1 ttl=53 time=34.502 ms
```

```
64 bytes from 114.215.133.69: seq=2 ttl=53 time=41.986 ms
^C
--- www.geeqee.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 33.405/36.631/41.986 ms
```

7.3 软件包管理 OPKG

在 OpenWrt 下使用的软件包管理技术叫做 OPKG，它是一个轻量级的软件包技术，已经成为开源嵌入式的标准之一，它类似 CentOS 下的 yum，可以实现对预编译的二进制软件进行下载。

7.3.1 OPKG 语法格式

```
opkg [参数 ...] 子命令 [子参数 ...]
参数有:
      update                        下载服务器上可用的软件包列表，由于软件包列表可
能会比较大，因此并不保存在系统中，每次启动需要首先执行 opkg update 取得最新的软件包。
      upgrade <包名>                升级软件包
      install <包名>                安装软件包
      configure <包名>              配置某一个软件包
      remove <包名>                 卸载软件包
      list                          列出全部可用的软件包
      list-installed                列出已安装的软件包
      list-upgradable               列出可以升级的软件包
      info [pkg|regexp]             显示出指定软件包的信息
      status [pkg|regexp]           显示出制定软件包的当前状态
      download <pkg>                下载一个软件包到当前文件夹（不安装）
      --verbosity[=<level>]         显示信息等级 :0 只显示错误，1 正常显示信息（默
认），2 显示需要的信息，3 调试，4 调试 2 级
      --force-overwrite             强制安装（谨慎使用）
      --force-remove                强制删除（谨慎使用）
```

每次开机后执行 OPKG 更新安装要更新的软件包。

```
root@OpenWrt:~# opkg update
Downloading http://www.maker1000.com/makerouter/system/ramips/packages/
base/Packages.gz.
Updated list of available packages in /var/opkg-lists/axwrt_base.
Downloading http://www.maker1000.com/makerouter/system/ramips/packages/
luci/Packages.gz.
wget: server returned error: HTTP/1.1 404 Not Found
Downloading
http://www.maker1000.com/makerouter/system/ramips/packages/management/
Packages.gz.
wget: server returned error: HTTP/1.1 404 Not Found
```

```
Downloading
http://www.maker1000.com/makerouter/system/ramips/packages/oldpackages/
Packages.gz
......
```

在执行过程中，会报几个错误，是因为有部分软件包暂不需要，因此库中没有。

7.3.2 软件包的内核依赖问题

在 OPKG 安装软件包的时候，有时会提示内核版本不匹配，大概如下所示。

```
Collected errors:
 * satisfy_dependencies_for: Cannot satisfy the following dependencies
for kmod-usb-storage:
 *      kernel (= 3.3.8-1-6acd2a17c333f503dc86081b03fe73c0) *   kernel
(= 3.3.8-1-6acd2a17c333f503dc86081b03fe73c0) *
 * opkg_install_cmd: Cannot install package kmod-usb-storage.
```

出现这个错误是因为服务器上的固件内核版本与设备上所安装的固件内核版本不一致，唯一的解决方法就是重新刷机。

7.3.3 OPKG 的配置

查看 OPKG 配置文件。


```
rroot@OpenWrt:/# cat /etc/opkg.conf
dest root /
dest ram /tmp
lists_dir ext /var/opkg-lists
option overlay_root /overlay
src/gz axwrt_base http://www.maker1000.com/makerouter/system/ramips/
packages/base
src/gz axwrt_luci http://www.maker1000.com/makerouter/system/ramips/
packages/luci
src/gz axwrt_management http://www.maker1000.com/makerouter/system/
ramips/packages/management
src/gz axwrt_oldpackages http://www.maker1000.com/makerouter/system/
ramips/packages/oldpackages
src/gz axwrt_packages http://www.maker1000.com/makerouter/system/
ramips/packages/packages
src/gz axwrt_routing http://www.maker1000.com/makerouter/system/ramips/
packages/routing
src/gz axwrt_telephony http://www.maker1000.com/makerouter/system/
ramips/packages/telephony
```

- dest root：安装目标根路径
- dest ram：内存临时文件路径

- lists_dir ext：软件包列表文件
- option overlay_root：可写分区挂载位置
- src/gz 下面的系列都是软件包分类获取地址

7.3.4 关于 OPKG 软件包

在未来的章节中，我们将会再次介绍 OPKG 相关内容，包括如何自己制作 OPKG 软件包。届时，读者可以更容易地将 x86 下的软件移植到 OpenWrt 中，也可以选择哪些软件包是要打包到固件的 ROM 中的，哪些是要动态安装的。

第8章 UCI 命令系统

8.1 UCI 命令

出于一个众所周知的原因，在 Linux 下各种软件包有各种不同的配置脚本，每个配置脚本的语法格式和操作方式都不同，这样的设计虽然可以体现出各软件包自身的优势，同时也增加了学习曲线。在这一点上，OpenWrt 的 UCI 无疑处理得更胜一筹。

UCI 是集中式配置信息管理接口（Unified Configuration Interface）的缩写，它是 OpenWrt 引进的一套配置参数管理系统。UCI 管理了 OpenWrt 下最主要的系统配置参数，并且提供了简单、容易、标准化的人机交互接口。UCI 中已经包含了网络配置、无线配置、系统信息配置等基本路由器所需的主要配置参数。同时，UCI 也可以帮助开发人员快速地建立一套基于 OpenWrt 的智能路由产品控制界面。

8.2 UCI 的文件和流程

UCI 的配置文件全部存储在 /etc/config 目录下。

```
root@OpenWrt:/# ls /etc/config/
dhcp      dropbear  firewall  network   system    wireless
```

日前已有大量软件包支持 UCI 模式管理，但不是所有的软件包都支持，支持的软件包是这样来完成启动的（以 samba 举例）：

（1）启动脚本 /etc/init.d/samba；

（2）启动脚本通过 UCI 分析库从 /etc/config/samba 获得启动参数；

（3）启动脚本完成正常启动。

由于 UCI 的数据文件较为简单，并且具备了很好的直接观感，所以配置文件既可以使用 UCI 命令进行修改，也可以使用 Vi 编辑器直接修改。但如果两种方式都使用，需要注意 UCI 命令修改会产生缓存，每次修改好要尽快确认保存，以免出现冲突。常见的几个 UCI 配置作用说明如表 8-1 所示。

表 8-1 常见 UCI 配置说明

文件	作用
/etc/config/dhcp	面向 LAN 口提供的 IP 地址分配服务配置
/etc/config/dropbear	SSH 服务配置
/etc/config/firewall	路由转发，端口转发，防火墙规则
/etc/config/network	自身网络接口配置
/etc/config/system	时间服务器时区配置
/etc/config/wireless	无线网络配置

8.3 UCI 的文件语法

UCI 文件语法举例：

```
config 'section-type' 'section'
        option   'key'       'value'
        list     'list_key'  'list_value'

config 'example' 'test'
        option   'string'      'some value'
        option   'boolean'     '1'
        list     'collection'  'first item'
        list     'collection'  'second item'
```

- config 节点：以关键字 config 开始的一行用来代表当前节点
  - section-type：节点类型
  - section：节点名称
- option 选项：表示节点中的一个元素
  - key：键
  - value：值
- list 列表选项：表示列表形式的一组参数
  - list_key：列表键
  - list_value：列表值

8.3.1 config 节点语法格式

```
config 'section-type' 'section'
```

config 节点（后文统一称为节点）原则：

- UCI 允许只有节点类型的匿名节点存在；

- 节点类型和名字建议使用单引号包含，以免引起歧义；
- 节点中可以包含多个 option 选项或 list 列表选项；
- 节点遇到文件结束或遇到下一个节点代表完成。

8.3.2 option 选项语法格式

```
option 'key' 'value'
```

option 选项（后文统一称为选项）原则：

- 选项的键与值建议使用单引号包含；
- 避免相同的选项键存在于同一个节点，否则只有一个生效。

8.3.3 list 列表选项语法格式

```
list 'list_key' 'list_value'
```

list 列表选项（后文统一称为列表）原则：

- 列表的键与值建议使用单引号包含；
- 列表键的名字如果相同，则相同键的值将会被当作数组传递给相应软件。

8.3.4 UCI 的语法容错

```
option example value
option 'example' value
option example "value"
option "example" 'value'
option 'example' "value"
```

8.3.5 UCI 无法容忍的语法

```
option 'example" "value'
option example some value with space
```

尽量使用常规字符去处理器 UCI，特殊字符有可能会破坏数据结构的完整性。

8.4 UCI 命令读写配置

语法格式如下。

```
uci [<options>] <command> [<arguments>]
```

读写规则：

- UCI 读取总是先读取内存中的缓存，然后再读取文件中的；
- 进行过增加、修改、删除操作后要执行生效指令，否则所做修改只留存在缓存中。

8.4.1 读取类语法

取得节点类型：

```
uci get <config>.<section>
```

取得一个值：

```
uci get <config>.<section>.<option>
```

显示全部 UCI 配置：

```
uci show
```

显示指定文件配置：

```
uci show <config>
```

显示指定节点名字配置：

```
uci show <config>.<section>
```

显示指定选项配置：

```
uci show <config>.<section>.<option>
```

显示尚未生效的修改记录：

```
uci changes <config>
```

匿名节点显示（如果所显示内容有匿名节点，使用 -X 参数可以显示出匿名节点的 ID）：

```
uci show -X <config>.<section>.<option>
```

8.4.2 写入类语法

在文件中增加一个匿名节点：

```
uci add <config> <section-type>
```

在文件中增加一个节点：

```
uci set <config>.<section>=<section-type>
```

在节点中增加一个选项和值：

```
uci set <config>.<section>.<option>=<value>
```

在列表中增加一个值：

```
uci add_list <config>.<section>.<option>=<value>
```

修改一个节点的类型：

```
uci set <config>.<section>=<section-type>
```

修改一个选项的值：

```
uci set <config>.<section>.<option>=<value>
```

删除指定名字的节点：

```
uci delete <config>.<section>
```

删除指定选项：

```
uci delete <config>.<section>.<option>
```

删除列表：

```
uci delete <config>.<section>.<list>
```

删除列表中一个值：

```
uci del_list <config>.<section>.<option>=<string>
```

生效修改（任何写入类的语法，最终都要执行生效修改，否则所做修改只在缓存中，切记！）：

```
uci commit <config>
```

8.5 综合实例

针对 UCI 命令，本书写了一篇综合实例，大家可以多做练习，不要害怕弄坏系统，万一真的弄坏了，重刷固件就行了。

8.5.1 语法测试

准备产生一个用来测试的配置 mytest。

```
root@OpenWrt:~# cp /etc/config/system /etc/config/mytest
root@OpenWrt:~# cat /etc/config/system
config system
        option hostname OpenWrt
        option timezone UTC

config timeserver ntp
        list server     0.openwrt.pool.ntp.org
        list server     1.openwrt.pool.ntp.org
        list server     2.openwrt.pool.ntp.org
        list server     3.openwrt.pool.ntp.org
        option enabled 1
        option enable_server 0
```

看看 mytest 的配置。

```
root@OpenWrt:~# uci show mytest
mytest.@system[0]=system
mytest.@system[0].hostname=OpenWrt
mytest.@system[0].timezone=UTC
mytest.ntp=timeserver
mytest.ntp.server=0.openwrt.pool.ntp.org 1.openwrt.pool.ntp.org
2.openwrt.pool.ntp.org 3.openwrt.pool.ntp.org
mytest.ntp.enabled=1
mytest.ntp.enable_server=0
```

取得 mytest.ntp.server 这个选项的值。

```
root@OpenWrt:~# uci get mytest.ntp.server
0.openwrt.pool.ntp.org 1.openwrt.pool.ntp.org 2.openwrt.pool.ntp.org
3.openwrt.pool.ntp.org
```

增加一个类型为 interface、名字为 hoowa 的节点。

```
uci set mytest.hoowa=interface
```

在 hoowa 节点中增加 enable 选项。

```
uci set mytest.hoowa.enable=1
```

修改类型为 system 的第一个匿名节点下的 hostname 选项。

```
uci set mytest.@system[0].hostname=smartrouter
```

删除掉列表 mytest.ntp.server 的一个值。

```
uci del_list mytest.ntp.server=2.openwrt.pool.ntp.org
```

查看都修改了什么。

```
root@OpenWrt:~# uci changes mytest
mytest.hoowa=interface
mytest.hoowa.enable=1
mytest.cfg02e48a.hostname=smartrouter
mytest.ntp.server-=2.openwrt.pool.ntp.org
```

使修改生效。

```
uci commit mytest
```

看看 mytest 文件。

```
root@OpenWrt:~# cat /etc/config/mytest

config system
        option timezone 'UTC'
        option hostname 'smartrouter'

config timeserver 'ntp'
        list server '0.openwrt.pool.ntp.org'
        list server '1.openwrt.pool.ntp.org'
        list server '3.openwrt.pool.ntp.org'
        option enabled '1'
        option enable_server '0'

config interface 'hoowa'
        option enable '1'
```


8.5.2 举例开启 Wi-Fi

OpenWrt 默认是不开启 Wi-Fi 的，这里带领大家打开 Wi-Fi。

查看当前 Wi-Fi 的配置。

```
root@OpenWrt:~# uci show wireless
wireless.radio0=wifi-device
wireless.radio0.type=mac80211
wireless.radio0.channel=11
wireless.radio0.hwmode=11g
wireless.radio0.path=10180000.wmac
wireless.radio0.htmode=HT20
```

```
wireless.radio0.disabled=1
wireless.@wifi-iface[0]=wifi-iface
wireless.@wifi-iface[0].device=radio0
wireless.@wifi-iface[0].network=lan
wireless.@wifi-iface[0].mode=ap
wireless.@wifi-iface[0].ssid=OpenWrt
wireless.@wifi-iface[0].encryption=none
```

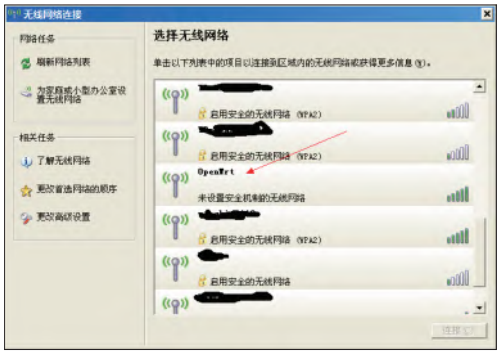
可以看到的是 wireless.radio0.disabled 为 1 表示禁用 Wi-Fi，打开方法如下所示。

```
root@OpenWrt:~# uci set wireless.radio0.disabled=0
root@OpenWrt:~# uci commit wireless
root@OpenWrt:~# uci show wireless.radio0.disabled
wireless.radio0.disabled=0
```

使用 Wi-Fi 命令启动。

```
root@OpenWrt:~# wifi
root@OpenWrt:~# [ 2976.140000] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 2976.140000] device wlan0 entered promiscuous mode
[ 2976.180000] br-lan: port 2(wlan0) entered forwarding state
[ 2976.180000] br-lan: port 2(wlan0) entered forwarding state
[ 2976.190000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 2978.180000] br-lan: port 2(wlan0) entered forwarding state
[ 2986.310000] device wlan0 left promiscuous mode
[ 2986.310000] br-lan: port 2(wlan0) entered disabled state
[ 2986.700000] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 2986.700000] device wlan0 entered promiscuous mode
[ 2986.710000] br-lan: port 2(wlan0) entered forwarding state
[ 2986.710000] br-lan: port 2(wlan0) entered forwarding state
[ 2986.750000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 2988.710000] br-lan: port 2(wlan0) entered forwarding state
```

如图 8-1 所示，扫描查看是否有一个叫 OpenWrt 的无线信号。



■ 图 8-1 扫描查看 OpenWrt 无线信号

第9章 智能路由配置

本章主要介绍智能路由器的配置，包括网络配置文件、WAN 口外网、LAN 口服务、无线网络、DHCP 服务配置。内容有一定难度，组合比较多，这是因为 OpenWrt 在网络部分设计得十分灵活。本章在文件编辑和测试的地方使用 Vi 编辑器或 UCI 命令两种方法来完成，两种方法的效果是相同的。

9.1 网络配置文件

路由器的基本功能是完成 WAN 口（连接互联网接口）、LAN 口（连接局域网接口）、Wireless（无线接口）的基本配置，这些配置都是以配置文件形式存在的，下面介绍几个配置文件。

- /etc/config/network：该文件是网络设备接口配置文件，可以对所有网口进行配置，还可以对 VLAN（虚拟局域网）进行配置。
- /etc/config/wireless：无线网络配置文件。该文件包含了无线网络协议、参数、速率等信息。

在 MR808 开发板上靠近 USB 接口一侧的网络接口就是网口，MT7620N 芯片一共提供了 5 个物理网口，分别为 ETHPHY0、ETHPHY1、ETHPHY2、ETHPHY3、ETHPHY4。在 CPU 看来这就是 5 个网口而已，至于它们为什么可以成为 WAN 和 LAN，那是因为使用了 VLAN 进行划分。

9.1.1 查看网络配置

```
root@OpenWrt:~# uci show network
network.loopback=interface
network.loopback.ifname=lo
network.loopback.proto=static
network.loopback.ipaddr=127.0.0.1
network.loopback.netmask=255.0.0.0
network.globals=globals
network.globals.ula_prefix=fdb4:329c:dc0b::/48
network.lan=interface
network.lan.ifname=eth0.1
network.lan.force_link=1
```

```
network.lan.type=bridge
network.lan.proto=static
network.lan.ipaddr=192.168.1.1
network.lan.netmask=255.255.255.0
network.lan.ip6assign=60
network.lan.macaddr=64:51:7e:33:98:44
network.wan=interface
network.wan.ifname=eth0.2
network.wan.proto=dhcp
network.wan.macaddr=64:51:7e:33:98:45
network.wan6=interface
network.wan6.ifname=@wan
network.wan6.proto=dhcpv6
```

- network.loopback：本地回环设备，如果在设备上访问 loopback 的 IP 地址，就是自己访问自己。
- network.lan：局域网设备。
- network.wan：外网设备。

9.1.2 查看当前网络

```
root@OpenWrt:~# ifconfig
br-lan    Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fdb4:329c:dc0b::1/60 Scope:Global
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:8834 (8.6 KiB)

eth0      Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:345 errors:0 dropped:0 overruns:0 frame:0
          TX packets:288 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60790 (59.3 KiB)  TX bytes:31302 (30.5 KiB)
          Interrupt:5

eth0.1    Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

```
          RX bytes:0 (0.0 B)  TX bytes:4152 (4.0 KiB)

eth0.2    Link encap:Ethernet  HWaddr 64:51:7E:33:98:45
          inet addr:192.168.0.170  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::6651:7eff:fe33:9845/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:307 errors:0 dropped:0 overruns:0 frame:0
          TX packets:239 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:50114 (48.9 KiB)  TX bytes:23919 (23.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7200 (7.0 KiB)  TX bytes:7200 (7.0 KiB)

wlan0     Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:3988 (3.8 KiB)
```

- br-lan：虚拟设备，LAN 口桥接设备，包含通过 LAN 口和 WAN 口连入系统的设备统一桥接。
- eth0：真实设备，CPU 内部到交换芯片之间只有一个接口。
- eth0.1：虚拟设备，由 VLAN 划分的有线的 LAN 口，VLAN 编号 1。
- eth0.2：虚拟设备，由 VLAN 划分的有线的 WAN 口，VLAN 编号 2。
- lo：虚拟设备，回环设备。
- wlan0：真实设备，启动 Wi-Fi 后将会产生此无线设备。
- pppoe-wan：虚拟设备，是 PPPOE 拨号上网成功后产生的。

9.1.3 查看 br-lan 桥状态

```
root@OpenWrt:~# brctl show
bridge name     bridge id               STP enabled     interfaces
br-lan          7fff.64517e339844       no              eth0.1
                                                        wlan0
```

9.1.4 查看内核日志

```
root@OpenWrt:~# logread
```

9.1.5 网口 VLAN

上文提到网口哪个是 WAN，哪个是 LAN，是由 VLAN 划分的，MR808 的划分是在两个地方进行的：一个是系统编译时的脚本文件，一个是 DTS 文件，在以后的章节中我们会专门讲这部分内容。

查看当前的 VLAN 划分情况。

```
root@OpenWrt:~# swconfig dev switch0 vlan 1 show
VLAN 1:
        vid: 1
        ports: 1 2 3 4 5 6t
root@OpenWrt:~# swconfig dev switch0 vlan 2 show
VLAN 2:
        vid: 2
        ports: 0 6t
```

其中 VLAN1 为 LAN 口的 VLAN，包含了 1、2、3、4 接口，后面的是交换机到 CPU 的内部接口。VLAN2 为 WAN 口，包含了 0 接口。

9.2 配置 WAN 口外网

WAN 口是任何一台路由器用来连接外网的端口，WAN 口可以被配置成为 3 种连接方式（实际上还可以有更多的方式，在以后的章节中会有介绍）。

/etc/config/network 配置文件内容如下。

```
config interface 'wan'
        option ifname 'eth0.2'
        option proto 'dhcp'
        option macaddr '64:51:7e:33:98:45'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show network.wan
network.wan=interface
network.wan.ifname=eth0.2
network.wan.proto=dhcp
network.wan.macaddr=64:51:7e:33:98:45
```

network.wan 选项参数说明如表 9-1 所示。

表 9-1　network.wan 选项参数说明

选项	说明	可选值及说明	必填
ifname	设备名称	eth0.2	是
proto	协议类型	static：静态 IP 地址 dhcp：动态获取 IP 地址 pppoe：拨号上网 pptp：远程 VPN 服务器 3g：连接 3G/4G 无线电话网络	是
macaddr	WAN 口 MAC 地址，修改该地址即可实现 MAC 地址克隆功能	首次数据根据 factory 分区内参数自动生成	是

9.2.1　动态获取 IP 选项

动态获取 IP 选项说明如表 9-2 所示。

表 9-2　动态获取 IP 选项说明

选项	说明	可选值及说明	必填
proto		dhcp	是
ifname		eth0.2	是
macaddr		根据 MT7620N 的 factory 分区自动生成的值	是
mtu	修改最大数据包大小，默认不用设置	数值	否
reqopts	在向 DHCP 服务器发出请求时增加附加的 DHCP 信息	字符串	否
dns	使用指定的 DNS 服务器地址替代获得的 DNS	字符串	否

配置文件举例如下。

```
config interface 'wan'
        option ifname 'eth0.2'
        option proto 'dhcp'
        option macaddr '00:11:22:33:44:55'
```

9.2.2　指定静态 IP 语法

指定静态 IP 语法说明如表 9-3 所示。

表 9-3　指定静态 IP 语法说明

选项	说明	可选值及说明	必填
proto		static	是
ifname		eth0.2	是
macaddr		根据 MT7620N 的 factory 分区自动生成的值	是

续表

选项	说明	可选值及说明	必填
mtu	修改最大数据包大小，默认不用设置	数值	否
ipaddr	WAN 口的 IP 地址	字符串	是
netmask	WAN 口的子网掩码	字符串	是
gateway	默认网关	字符串	否
broadcast	广播地址	字符串	否
dns	DNS 服务器地址	字符串	否

配置文件举例如下。

```
config interface 'wan'
        option ifname 'eth0.2'
        option proto 'static'
        option ipaddr '192.168.0.2'
        option netmask '255.255.255.0'
        option gateway '192.168.0.1'
        option dns '192.168.0.1'
```

9.2.3 PPPOE 拨号上网语法

在设置 pppoe 时要确保你的设备 WAN 口连接的外网具备 PPPOE 的服务器，比如小区宽带、光猫或上联 ADSL 猫。PPPOE 拨号上网语法说明如表 9-4 所示。

表 9-4 PPPOE 拨号上网语法说明

选项	说明	可选值及说明	必填
proto		pppoe	是
ifname		eth0.2	是
macaddr		根据 MT7620N 的 factory 分区自动生成的值	是
mtu	修改最大数据包大小，默认不用设置	数值	否
username	拨号用的账号	字符串	是
password	拨号用的密码	字符串	是
ac	使用指定的访问集中器进行连接	字符串	否
service	连接的服务名称	字符串	否
connect	连接时候执行的外部脚本	字符串	否
disconnect	断开连接时执行的外部脚本	字符串	否
demand	等待多久没有活动就断开 PPPOE 连接	数字，单位为秒	否
dns	DNS 服务器地址	字符串	否
pppd_options	用于 pppd 进程执行时候的附加参数	字符串	否

配置文件举例如下。

```
config interface 'wan'
        option ifname 'eth0.2'
        option proto 'pppoe'
        option username '280000000000'
        option password '21700000'
```

9.2.4 WAN 口设置

使用 Vi 编辑器修改 WAN 口配置。

（1）编辑 /etc/config/network 文件。

```
root@OpenWrt:~# vi /etc/config/network
```

（2）删除掉 wan 节点中所有选项，只保留 macaddr。

（3）新增如下信息。

```
config interface 'wan'
        option ifname 'eth0.2'
        option proto 'dhcp'
        option macaddr '第二步保留的信息'
```


（4）保存退出编辑模式。

（5）重启网络使配置生效。

```
root@OpenWrt:~# /etc/init.d/network restart
```

9.2.5 WAN 口设置

（1）将运营商的网线接入 MR808 开发板的 WAN 口，假设账号为 280000000000，密码为 21700000。

（2）查看当前 WAN 口的 macaddr，并且记录到临时文件中。

```
root@OpenWrt:~# uci show network.wan.macaddr > /tmp/macaddr
```

（3）删除掉旧的 network.wan。

```
root@OpenWrt:~# uci delete network.wan
```

（4）写入新的 WAN 口配置。

```
root@OpenWrt:~# uci set network.wan=interface
```

```
root@OpenWrt:~# uci set network.wan.ifname='eth0.2'
root@OpenWrt:~# uci set network.wan.proto='pppoe'
root@OpenWrt:~# uci set network.wan.username='280000000000'
root@OpenWrt:~# uci set network.wan.password='21700000'
```

（5）查看刚才保存的 network.wan.macaddr 信息，并且填回（屏幕中的 64:xx:xx:xx:xx:xx 只是本书自己的 MR808 开发板的 macaddr，每个板子都不一样）。

```
root@OpenWrt:~# cat /tmp/macaddr
network.wan.macaddr=64:51:7e:33:98:45
root@OpenWrt:~# uci set network.wan.macaddr='64:51:7e:33:98:45'
```

（6）查看修改，确认是否跟自己想要的一样。

```
root@OpenWrt:~# uci changes
-network.wan
network.wan=interface
network.wan.ifname=eth0.2
network.wan.proto=pppoe
network.wan.username=280000000000
network.wan.password=21700000
network.wan.macaddr=64:51:7e:33:98:45
```

（7）保存生效。

```
root@OpenWrt:~# uci commit network
```

（8）重启网络，使配置生效。

```
root@OpenWrt:~# /etc/init.d/network restart
```

（9）重启后一般拨号要等待 30 秒 ~5 分钟，如果参数没错，就会拨号成功。

（10）查看拨号后是否建立了设备。

```
root@OpenWrt:~# ifconfig pppoe-wan
```

9.3 配置 LAN 口服务

LAN 口下的设备可以通过 WAN 口接入网络，也可以直接访问设备上的各项功能（系统防火墙对 LAN 口默认不做任何拦截）。network.lan 选项参数说明如表 9-5 所示。

/etc/config/network 配置文件内容如下。

```
config interface 'lan'
        option ifname 'eth0.1'
```

```
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60'
        option macaddr '64:51:7e:33:98:44'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show network.lan
network.lan=interface
network.lan.ifname=eth0.1
network.lan.force_link=1
network.lan.type=bridge
network.lan.proto=static
network.lan.ipaddr=192.168.1.1
network.lan.netmask=255.255.255.0
network.lan.ip6assign=60
network.lan.macaddr=64:51:7e:33:98:44
```

表 9-5　network.lan 选项参数说明

选项	说明	可选值及说明	必填
ifname	设备名称，一旦改了就乱了	eth0.1	是
proto	协议类型	static：静态 IP 地址	是
macaddr	WAN 口 MAC 地址，修改该地址即可实现 MAC 地址克隆功能	首次数据根据 factory 分区内参数自动生成	是
type	网络类型必须是桥模式，要不就没交换机功能了	bridge	是
ipaddr	LAN 口的 IP 地址，用于局域网内其他设备访问路由器	字符串	是
netmask	子网掩码	字符串	是

配置文件举例如下。

```
config interface 'lan'
        option ifname 'eth0.1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
```

修改过 LAN 口的配置后要重启网络以及 DHCP 服务。

```
root@OpenWrt:~# /etc/init.d/network restart
root@OpenWrt:~# /etc/init.d/odhcpd restart
```

9.4 配置无线网络

在 MR808 开发板的核心板上已经内置了天线，不必外接天线即可实现无线功能。/etc/config/wireless 配置文件内容如下。

```
config wifi-device  radio0
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     '10180000.wmac'
        option htmode   HT20
        # REMOVE THIS LINE TO ENABLE WIFI:
        option disabled 1

config wifi-iface
        option device   radio0
        option network  lan
        option mode     ap
        option ssid     OpenWrt
        option encryption none
```

UCI 查看方法如下。

```
root@OpenWrt:/etc/config# uci show wireless
wireless.radio0=wifi-device
wireless.radio0.type=mac80211
wireless.radio0.channel=11
wireless.radio0.hwmode=11g
wireless.radio0.path=10180000.wmac
wireless.radio0.htmode=HT20
wireless.radio0.disabled=1
wireless.@wifi-iface[0]=wifi-iface
wireless.@wifi-iface[0].device=radio0
wireless.@wifi-iface[0].network=lan
wireless.@wifi-iface[0].mode=ap
wireless.@wifi-iface[0].ssid=OpenWrt
wireless.@wifi-iface[0].encryption=none
```

其中 wireless.radio0 选项的参数说明如表 9-6 所示。

表 9-6　wireless.radio0 选项参数说明

选项	说明	可选值及说明	必填
type	设备类型	mac80211	是
channel	无线信道，不同国家类型所支持的信道不同	auto、1、2、3、4、5、6、7、8、9、10、11、12、13	是
hwmode	无线协议类型	11g：表示 802.11g+n	是
path	无线配置信息路径	10180000.wmac	是
htmode	无线频宽	HT20 HT40	是
disabled	关闭无线设备	0：禁用 1：启用	是
country	国家类型，跟支持频道有关，中国为 CN，支持 1 ~ 13	CN：中国	否
txpower	无线发射功率	0~20，单位为 dBm	否

配置文件举例如下。

```
config wifi-device  radio0
        option type     'mac80211'
        option channel  '11'
        option hwmode   '11g'
        option path     '10180000.wmac'
        option htmode   'HT20'
        option disabled '0'
        option country 'CN'
```

wireless.@wifi-iface[0] 选项的参数说明如表 9-7 所示。

表 9-7　wireless.@wifi-iface[0] 选项参数说明

选项	说明	可选值及说明	必填
device	关联无线设备	radio0	是
network	关联网络设备类型	lan：表示桥接到 LAN 网上 wwan：表示启用无线中继	是
mode	无线工作模式	ap：访问点模式 sta：客户端模式	是
ssid	无线的名称，尽量用英文	字符串	是
hidden	隐藏无线名称	0：禁用 1：启用	否
wmm	无线多媒体优先支持	0：禁用 1：启用	否
encryption	无线加密方式，推荐使用 psk-mixed 模式	none：不加密 psk：WPA-PSK 模式 psk2：WPA-PSK2 模式 psk-mixed：WPA-PSK / WPA-PSK2 混合	是
key	无线密钥，只有设置了无线加密才有效	字符串，长度为 8~64 个 ASCII 字符	否

配置文件举例如下。

```
config wifi-iface
        option device   'radio0'
        option network  'lan'
        option mode     'ap'
        option ssid     'OpenWrt'
        option encryption 'psk-mixed'
        option key      '12345678'
```

修改过无线配置后需要生效。

```
root@OpenWrt:~# wifi
```

9.4.1 无线查看命令

查看无线网络状态。

```
root@OpenWrt:~# iwinfo
wlan0     ESSID: "OpenWrt"
          Access Point: 64:51:7E:33:98:44
          Mode: Master  Channel: 11 (2.462 GHz)
          Tx-Power: 20 dBm  Link Quality: unknown/70
          Signal: unknown  Noise: unknown
          Bit Rate: unknown
          Encryption: mixed WPA/WPA2 PSK (CCMP)
          Type: nl80211  HW Mode(s): 802.11bgn
          Hardware: unknown [Generic MAC80211]
          TX power offset: unknown
          Frequency offset: unknown
          Supports VAPs: yes  PHY name: phy0
```

显示在左面的 wlan0 为无线连接的设备名字，这个名字是由匿名节点 wireless.@wifi-iface[0] 创建的。

通过 wlan0 搜索范围内其他无线设备。

```
root@OpenWrt:~# iwinfo wlan0 scan
```

通过 wlan0 查看 radio0 支持的国家类型编码。

```
root@OpenWrt:~# iwinfo wlan0 countrylist
```

9.4.2 实现无线中继

无线中继可以实现以 MR808 开发板为客户端远程连接另外一个无线路由器，主要是

通过增加 wifi-iface 实现的。增加后，系统就有两个匿名的 wifi-iface 配置，其中一个用来解决其他设备接入路由器，另外一个用来解决无线中继。

为网络增加 WWAN 类型接口设备。

```
root@OpenWrt:~# uci set network.wwan=interface
root@OpenWrt:~# uci set network.wwan.proto=dhcp
root@OpenWrt:~# uci commit
root@OpenWrt:~# uci show network.wwan
network.wwan=interface
network.wwan.proto=dhcp
```

将 WWAN 设置到防火墙 WAN 区域范围。

由于 WAN 区域属于匿名节点，首先找到 WAN 口节点的 ID 号（下面示例中 cfg06dc81 为节点 ID，这个值是随机的，每台机器操作时可能有所不同）。

```
root@OpenWrt:~# uci show -X firewall|grep name=wan
firewall.cfg06dc81.name=wan
root@OpenWrt:~# uci show firewall.cfg06dc81.network
firewall.cfg06dc81.network=wan wan6
```

修改区域范围。


```
root@OpenWrt:~# uci set firewall.cfg06dc81.network='wan wan6 wwan'
root@OpenWrt:~# uci commit
```

查看是否修改成功。

```
root@OpenWrt:~# uci show firewall.cfg06dc81
firewall.cfg06dc81=zone
firewall.cfg06dc81.name=wan
firewall.cfg06dc81.input=REJECT
firewall.cfg06dc81.output=ACCEPT
firewall.cfg06dc81.forward=REJECT
firewall.cfg06dc81.masq=1
firewall.cfg06dc81.mtu_fix=1
firewall.cfg06dc81.network=wan wan6 wwan
```

新建 wifi-iface 匿名节点实现连接。

（1）先搜索你要中继的另外一个无线是否能搜索到，举例如下。

```
root@OpenWrt:~# iwinfo wlan0 scan
Cell 01 - Address: AC:A2:13:93:19:D8
          ESSID: "FREEIRIS"
          Mode: Master  Channel: 11
          Signal: -56 dBm  Quality: 54/70
```

```
          Encryption: mixed WPA/WPA2 PSK (CCMP)
```

（2）根据获得的值向配置文件 /etc/config/wireless（使用 Vi 或 UCI 两种方式都可以）中增加一个类型为 wifi-iface 的匿名节点，其中 ssid 填写获得的 ESSID，bssid 填写获得的 Address，而 Encryption 要和对方对应，我们支持 psk、psk2、psk-mixed 三种类型方式，举例如下。

```
config wifi-iface
        option device   'radio0'
        option network  'wwan'
        option mode     'sta'
        option ssid     'FREEIRIS'
        option bssid    'AC:A2:13:93:19:D8'
        option encryption 'psk-mixed'
        option key      '12345678'
```

（3）使用 wifi 命令生效设置，之后检测，如果发现 wlan0 和 wlan0-1 两个设备，则表示生效，其中一个为本机的无线服务，另外一个表示远程无线连接到其他路由器上，如果这里连接失败，肯定是配置有问题，请重复上面前两步操作。

```
root@OpenWrt:~# wifi
root@OpenWrt:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 64:51:7E:33:98:44
          inet addr:192.168.199.226  Bcast:192.168.199.255
Mask:255.255.255.0
          inet6 addr: fe80::6651:7eff:fe33:9844/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:35 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7676 (7.4 KiB)  TX bytes:4448 (4.3 KiB)

root@OpenWrt:~# ifconfig wlan0-1
wlan0-1   Link encap:Ethernet  HWaddr 64:51:7E:33:98:45
          inet6 addr: fe80::6651:7eff:fe33:9845/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1024 (1.0 KiB)
```

- 无线中继要修改的文件较多，请认真操作，如果遇到错误需要重复进行。
- 无线中继建立成功后所具备的功能如下所示。
- ifconfig 命令可以看到多了两个连接：一个是 wlan0，另外一个是 wlan0-1，其中有一个带 IP 地址，带 IP 地址的就是无线中继。

- MR808 开发板本身可以直接连接外网。
- 通过无线或 LAN 口连接到 MR808 开发板上的手机或计算机可以通过 MR808 开发板的无线中继连接外网。
- 无线连接失败：无线连接失败多半是由于配置信息写错引起的，重新检查重复操作步骤“新建 wifi-iface 匿名节点实现连接”。
- 如何关闭无线访问点模式：直接去掉另外那个 wifi-iface，并且重启无线网络就可以了。

9.5 DHCP 服务

OpenWrt 的基础软件包中包含了一个名为 dnsmasq 的 DHCP 服务，可以为 LAN 下的设备提供自动分配 IP 地址。默认的配置已经可以完全达到服务要求了，因此如果不想修改可以暂时跳过本节。

/etc/config/dhcp 配置文件内容如下。

```
config dnsmasq
        option domainneeded '1'
        option boguspriv '1'
        option filterwin2k '0'
        option localise_queries '1'
        option rebind_protection '1'
        option rebind_localhost '1'
        option local '/lan/'
        option domain 'lan'
        option expandhosts '1'
        option nonegcache '0'
        option authoritative '1'
        option readethers '1'
        option leasefile '/tmp/dhcp.leases'
        option resolvfile '/tmp/resolv.conf.auto'

config dhcp 'lan'
        option interface 'lan'
        option start '100'
        option limit '150'
        option leasetime '12h'
        option dhcpv6 'server'
        option ra 'server'

config dhcp 'wan'
        option interface 'wan'
        option ignore '1'

config odhcpd 'odhcpd'
        option maindhcp '0'
```

```
        option leasefile '/tmp/hosts/odhcpd'
        option leasetrigger '/usr/sbin/odhcpd-update'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show dhcp
dhcp.@dnsmasq[0]=dnsmasq
dhcp.@dnsmasq[0].domainneeded=1
dhcp.@dnsmasq[0].boguspriv=1
dhcp.@dnsmasq[0].filterwin2k=0
dhcp.@dnsmasq[0].localise_queries=1
dhcp.@dnsmasq[0].rebind_protection=1
dhcp.@dnsmasq[0].rebind_localhost=1
dhcp.@dnsmasq[0].local=/lan/
dhcp.@dnsmasq[0].domain=lan
dhcp.@dnsmasq[0].expandhosts=1
dhcp.@dnsmasq[0].nonegcache=0
dhcp.@dnsmasq[0].authoritative=1
dhcp.@dnsmasq[0].readethers=1
dhcp.@dnsmasq[0].leasefile=/tmp/dhcp.leases
dhcp.@dnsmasq[0].resolvfile=/tmp/resolv.conf.auto
dhcp.lan=dhcp
dhcp.lan.interface=lan
dhcp.lan.start=100
dhcp.lan.limit=150
dhcp.lan.leasetime=12h
dhcp.lan.dhcpv6=server
dhcp.lan.ra=server
dhcp.wan=dhcp
dhcp.wan.interface=wan
dhcp.wan.ignore=1
dhcp.odhcpd=odhcpd
dhcp.odhcpd.maindhcp=0
dhcp.odhcpd.leasefile=/tmp/hosts/odhcpd
dhcp.odhcpd.leasetrigger=/usr/sbin/odhcpd-update
```

其中 dhcp.@dnsmasq[0] 选项的参数说明如表 9-8 所示。dhcp.lan 选项的参数说明如表 9-9 所示。dhcp.wan 选项的参数说明如表 9-10 所示。

表 9-8　dhcp.@dnsmasq[0] 选项参数说明

选项	说明	可选值及说明	必填
domainneeded	不转发无效格式域名	0：禁用 1：启用	是
boguspriv	不转发私有地址空间	0：禁用 1：启用	是

续表

选项	说明	可选值及说明	必填
filterwin2k	微软的特殊请求包处理	0：禁用 1：启用	是
localise_queries	允许获得同时发生的请求的来源网络地址	0：禁用 1：启用	是
rebind_protection	拒绝来自上游服务器并且属于本地私有 IP 段的绑定请求	0：禁用 1：启用	是
rebind_localhost	本机网段的绑定请求允许	0：禁用 1：启用	是
local	设定本地网段	/lan/：来自 LAN 局域网	是
domain	设定本地网域名	lan	是
expandhosts	在 /etc/hosts 文件中给域名增加简单的名字	0：禁用 1：启用	是
nonegcache	禁用负极缓存	0：禁用 1：启用	是
authoritative	dhcp 认证	0：禁用 1：启用	是
readethers	读取 /etc/ethers 文件中关于 hosts 的信息	0：禁用 1：启用	是
leasefile	存储客户端 DHCP 申请信息的文件名	字符串	是
resolvfile	上游 DNS 地址文件名	字符串	是

表 9-9　dhcp.lan 选项参数说明

选项	说明	可选值及说明	必填
interface	对应网络设备	lan	是
start	分配 IP 地址的开始位置	数字，范围为 0~255	是
limit	分配 IP 地址总量	数字，加上开始位置，不能超过 255	是
leasetime	DHCP 租期	数字时间，12h 表示 12 个小时	是

表 9-10 dhcp.wan 选项参数说明

选项	说明	可选值及说明	必填
interface	对应网络设备	wan	是
ignore	忽略这个网络设备	0：禁用 1：启用	是

修改过后要重启 DHCP 服务。

```
root@OpenWrt:~# /etc/init.d/dhcp restart
```

9.6 如何连接外网

我们很是好奇，在看本书的时候你是怎么让 MR808 开发板连接外网的？这里给出一些连接外网的方式，如表 9-11 所示，看看你达到哪个级别了。

表 9-11 MR808 开发板连接外网方式

级别	模型
小猫	计算机→ Wi-Fi →MR808 开发板→（WAN 口）→外网路由器→外网
老虎	计算机→ Wi-Fi →MR808 开发板→（WAN 口）→外网
喵星人	计算机→ Wi-Fi →外网路由器→外网 计算机→ LAN（不设网关）→ MR808 开发板→（WAN 口）→外网路由器→外网
喵了个咪	计算机→ Wi-Fi →MR808 开发板→（无线中继）→外网路由器→外网

第10章 防火墙与 UPnP

10.1 防火墙命令

言归正传，防火墙、DMZ 在 OpenWrt 系统中都是由 /etc/config/firewall 配置文件管理的，并且 NAT 转发等很多东西也是由该文件管理的。使用该文件的优点是，不用再学复杂的 iptables 命令了，让我们跟 iptables say good bye 吧!

Tips

Firewall：防火墙。
DMZ：独立隔离区，处在独立隔离区的计算机是直接暴露在外，不经过防火墙保护的。
UPnP：一种多媒体传输协议，用于游戏、娱乐软件。

重置防火墙命令。

```
root@OpenWrt:~# /etc/init.d/firewall reload
```

重启防火墙命令。

```
root@OpenWrt:~# /etc/init.d/firewall restart
```

查看防火墙完整策略。

```
root@OpenWrt:~# iptables -L
```

10.2 防火墙配置

防火墙文件是 OpenWrt 中一个比较大的 UCI 文件，其内容几乎全部是匿名的配置节点，这让配置变得稍显麻烦，建议使用 UCI 的 -X 参数查看或使用 Vi 编辑器查看防火墙文件。

```
root@OpenWrt:~# uci show -X firewall
```

为了降低学习难度，本书将防火墙文件的内容分为 5 类节点。

- defaults 类型匿名节点：只有一个配置节点，默认配置。
- zone 类型匿名节点：系统将 LAN 和 WAN 分为两个不同的 zone，两个 zone 之间是

隔离的。

- forwarding 类型匿名节点：用于不同 zone 之间的转发。
- rule 类型匿名节点：用来进行配置哪些端口允许访问。
- direction 类型匿名节点：用来配置实现具体端口转发功能。

10.2.1 defaults 默认配置

firewall.defaults 虽是匿名节点，但它只有一份，不会出现多个。

配置内容如下。

```
config defaults
        option syn_flood '1'
        option input 'ACCEPT'
        option output 'ACCEPT'
        option forward 'REJECT'
```

firewall.@defaults 选项的参数说明如表 10-1 所示。

表 10-1 firewall.@defaults 选项参数说明

选项	说明	可选值及说明	必填
input	INPUT 链过滤策略	ACCEPT：允许 REJECT：拒绝	是
output	OUTPUT 链过滤策略	ACCEPT：允许 REJECT：拒绝	是
forward	FORWARD 链过滤策略	ACCEPT：允许 REJECT：拒绝	是
drop_invalid	抛弃无效数据包	0：禁用 1：启用	否
syn_flood	是否启用防洪水攻击	0：禁用 1：启用	否
synflood_rate	设置检测 SYN 包阈值，默认值为 25	数字，单位为包 / 秒	否
synflood_burst	设置检测 SYN 包为洪水攻击阈值，默认值为 50	数字，单位为包 / 秒	否
disable_ipv6	关闭 IPv6 防火墙策略，默认为 0	0：禁用 1：启用	否

10.2.2 zone 域配置

在 OpenWrt 中，采用域类型的节点主要用来描述 WAN，另外一个用来描述 LAN。一般情况下，除非特别复杂的网络，否则不需要创建新的 zone。

配置内容如下。

```
config zone
        option name 'lan'
        list network 'lan'
        option input 'ACCEPT'
        option output 'ACCEPT'
        option forward 'ACCEPT'

config zone
        option name 'wan'
        option input 'REJECT'
        option output 'ACCEPT'
        option forward 'REJECT'
        option masq '1'
        option mtu_fix '1'
        list network 'wan'
        list network 'wan6'
        list network 'wwan'
```

firewall.@zone 选项的参数说明如表 10-2 所示。

表 10-2　firewall.@zone 选项参数说明

选项	说明	可选值及说明	必填
name	名字，用来区分 zone 匿名节点	字符串	是
network	列表值，什么接口设备被绑定到这个 zone 上	字符串，一般是网络接口设备名称	否
input	默认情况下输入传输策略	ACCEPT：允许 REJECT：拒绝	否
output	默认情况下输出传输策略	ACCEPT：允许 REJECT：拒绝	否
forward	默认情况下转发传输策略	ACCEPT：允许 REJECT：拒绝	否
masq	传输伪装开关，如是 WAN 口，必须设为 1	0：禁用 1：启用	否
mtu_fix	数据输出时开启 MSS 钳制，如果是 WAN 口，请设为 1	0：禁用 1：启用	否

10.2.3　forwarding 转发配置

通过转发配置可以实现两个不同 zone 域之间的数据发送，除非特别复杂的网络，否则无需创建或修改转发配置。

配置内容如下。

```
config forwarding
        option src 'lan'
        option dest 'wan'
```

firewall.@forwarding 选项的参数说明如表 10-3 所示。

表 10-3 firewall.@forwarding 选项参数说明

选项	说明	可选值及说明	必填
src	来源 zone	字符串	是
dest	目标 zone	字符串	是
family	IP 协议类型，默认为 any	any：任何协议	否

10.2.4 rule 规则

rule 规则是防火墙的许可规则机制，它是之前介绍过的 5 大类节点之一，rule 规则可以有很多内容，本书在 WAN 的 zone 配置中介绍过，进入所有 WAN 口的请求都会被拒绝，如果希望有例外，那么例外要通过 rule 来实现许可。

任何一个 IP 数据包都存在 src 来源、dest 目标。firewall.@rule 选项参数说明如表 10-4 所示。

表 10-4 firewall.@rule 选项参数说明

选项	说明	可选值及说明	必填
name	名字，用来区分 rule 匿名节点	字符串	是
target	规则动作	ACCEPT：允许 REJECT：拒绝 DROP：抛弃	是
src	数据源的 zone 域	字符串	否
src_ip	数据源的 IP 地址	字符串，IP 地址	否
src_mac	数据源的 MAC 地址	字符串，MAC 地址	否
src_port	数据源的端口，可以是一个端口或是端口范围，但必须同时指定参数 proto	数字，端口编号 字符串，端口范围，如 1000~2000	否
proto	数据源的协议类型	all：表示任意协议 具体协议：tcp、udp、tcpudp、udplit、icmp、esp、ah、sctp	否

续表

选项	说明	可选值及说明	必填
dest	目的地的 zone 域	字符串	否
dest_ip	目的地的 IP 地址	字符串，IP 地址	否
dest_port	目的地的端口，可以是一个端口或是端口范围，但必须同时指定参数 proto	数字，端口编号 字符串，端口范围，如 1000~2000	否
family	IP 协议类型，默认为 any	字符串，ipv4、ipv6、any	否

举例，开启 WAN 口的 ping 响应。

```
config rule
        option name 'Allow-Ping'
        option src 'wan'
        option proto 'icmp'
        option icmp_type 'echo-request'
        option family 'ipv4'
        option target 'ACCEPT'
```

10.2.5 direction 端口转发

端口转发是路由器中常见的功能，它允许访问者通过 WAN 口访问 LAN 口中的一个特定端口，并且将结果转发回给访问者。例如，将 80 端口开放到 WAN 口上。转发是常用的路由器配置参数，因此允许多个匿名段存在。firewall.@direction 选项的参数说明如表 10-5 所示。

表 10-5 firewall.@direction 选项参数说明

选项	说明	可选值及说明	必填
name	名字，用来区分 direction 匿名节点	字符串	是
src	被转发来源 zone 域	字符串，一般是 wan	是
src_ip	被转发的 IP 地址	字符串，IP 地址	否
src_mac	被转发的 MAC 地址	字符串，MAC 地址	否
src_port	被转发的端口	数字，端口编号	否
proto	协议类型	all：表示任意协议 具体协议：tcp、udp、tcpudp、udplit、icmp、esp、ah、sctp	否

续表

选项	说明	可选值及说明	必填
dest	转发到哪个 zone 域	字符串	否
dest_ip	转发到哪个 IP 地址	字符串，IP 地址	否
dest_port	转发到哪个端口	数字，端口编号	否
dest_mac	转发到哪个 MAC 地址	字符串，MAC 地址编号	否

举例，所有来自 WAN 口的 TCP 协议访问 9020 的请求都转发给局域网中一台 80 端口的计算机。

```
config redirect
        option name 'mytest'
        option proto 'tcp'
        option src 'wan'
        option src_dport '9020'
        option dest 'lan'
        option dest_ip '192.168.1.100'
        option dest_port '80'
```

举例，将 IP 地址 192.168.1.2 设置到 DMZ 隔离区。

```
config redirect
      option src              wan
      option proto            all
      option dest_ip          192.168.1.2
```

10.3 实现 UPnP

UPnP 是一种对等即插即用网络协议，主要用于视频、音频领域的传输，对使用者来说，打开 UPnP 之后可以增加迅雷等下载软件的下载速度。如果开启了 UPnP，会消耗一定 CPU 资源和内存，请谨慎考虑！

10.3.1 安装 UPnP

OpenWrt 下的 UPnP 服务端软件名为 miniupnpd，该软件不属于基本软件包，需要在线下载，请确保 MR808 开发板可以连接上外网，然后进行下面的步骤来下载、安装。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install miniupnpd
```

10.3.2 配置 UPnP

配置内容如下。

```
root@OpenWrt:~# cat /etc/config/upnpd
config upnpd config
        option enable_natpmp    1
        option enable_upnp      1
        option secure_mode      1
        option log_output       0
        option download         1024
        option upload           512
        #by default, looked up dynamically from ubus
        #option external_iface  wan
        option internal_iface   lan
        option port             5000
        option upnp_lease_file  /var/upnp.leases

config perm_rule
        option action           allow
        option ext_ports        1024-65535
        option int_addr         0.0.0.0/0    # Does not override secure_mode
        option int_ports        1024-65535
        option comment          "Allow high ports"

config perm_rule
       option action            deny
       option ext_ports         0-65535
       option int_addr          0.0.0.0/0
       option int_ports         0-65535
       option comment           "Default deny"
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show upnpd
upnpd.config=upnpd
upnpd.config.enable_natpmp=1
upnpd.config.enable_upnp=1
upnpd.config.secure_mode=1
upnpd.config.log_output=0
upnpd.config.download=1024
upnpd.config.upload=512
upnpd.config.internal_iface=lan
upnpd.config.port=5000
upnpd.config.upnp_lease_file=/var/upnp.leases
upnpd.@perm_rule[0]=perm_rule
upnpd.@perm_rule[0].action=allow
```

```
upnpd.@perm_rule[0].ext_ports=1024-65535
upnpd.@perm_rule[0].int_addr=0.0.0.0/0
upnpd.@perm_rule[0].int_ports=1024-65535
upnpd.@perm_rule[0].comment=Allow high ports
upnpd.@perm_rule[1]=perm_rule
upnpd.@perm_rule[1].action=deny
upnpd.@perm_rule[1].ext_ports=0-65535
upnpd.@perm_rule[1].int_addr=0.0.0.0/0
upnpd.@perm_rule[1].int_ports=0-65535
upnpd.@perm_rule[1].comment=Default deny
```

其中 upnpd.config 选项的参数说明如表 10-6 所示。

表 10-6 upnpd.config 选项参数说明

选项	说明	可选值及说明	必填
enable_natpmp	开启 NAT-PMP 支持	0：禁用 1：启用	是
enable_upnp	开启 UPnP 支持	0：禁用 1：启用	是
secure_mode	安全模式，客户端只能给自己转发一个输入口	0：禁用 1：启用	是
log_output	日志输出级别，0 表示不输出日志，如果设置了，将输出到 syslog 中	数字，范围为 0~5	是
download	允许来自 WAN 口的数据输入带宽	数字，单位为 KB/s	是
upload	允许输出到 WAN 口的数据输入带宽	数字，单位为 KB/s	是
external_iface	外网的 zone 域，默认为 WAN	字符串	是
internal_iface	内网的 zone 域，默认为 WAN	字符串	是
port	服务监听端口	数字	是
upnp_lease_file	UPnP 客户端租用记录文件路径	字符串，文件路径	是

perm_rule 为许可匿名配置节点，允许多个匿名配置节点同时存在，主要进行端口授权配置，行为类似于防火墙的 rule 规则。

upnpd.@perm_rule 选项的参数说明如表 10-7 所示。

表 10-7　upnpd.@perm_rule 选项参数说明

选项	说明	可选值及说明	必填
action	动作许可	allow：许可 deny：不许可	是
ext_ports	外部端口范围	字符串，开始端口 – 结束端口，例如 0~65535	是
int_addr	IP 地址，如果是 0.0.0.0/0 表示全部	字符串，IP 地址	是
int_ports	内部端口范围	字符串，开始端口 – 结束端口，例如 0~65535	是
comment	备注信息，没有实际作用	字符串	是

举例，禁止低于 1024 的端口被 UPnP 客户端访问。

```
config perm_rule
        option action           allow
        option ext_ports        1024-65535
        option int_addr         0.0.0.0/0    # Does not override secure_mode
        option int_ports        1024-65535
        option comment          "Allow high ports"

config perm_rule
       option action            deny
       option ext_ports         0-65535
       option int_addr          0.0.0.0/0
       option int_ports         0-65535
       option comment           "Default deny"
```


10.3.3　UPnP 命令

设置开机自动启动。

```
root@OpenWrt:~# /etc/init.d/miniupnpd enable
```

临时启动 miniupnpd。

```
root@OpenWrt:~# /etc/init.d/miniupnpd start
```

查看 miniupnpd 是否驻留内存。

```
root@OpenWrt:/etc/config# netstat -lnp|grep miniupnpd
```

第11章 时区、服务、SSH 与 SCP

11.1 信息和时区

大家知道计算机重新开机后时间仍是实时准确的，那是因为主板上有电池和时钟芯片，一般该芯片是达拉斯品牌的。在路由器上通常不会有时钟芯片和电池（传统意义上的路由器也不需要），那些路由器只提供路由功能，时间没有什么用处。可是智能路由器中可能需要这个功能，至少如果有数据库，数据库记录数据的时间总是要正确的。

在不能改变硬件的情况下解决这个问题，需要采用 NTP 协议向网络上的时间授权服务器请求获得时间服务，再根据本地时区的配置转换为本地时间。

NTP（Network Time Protocol，网络时间协议）是用来使网络中的各个计算机时间同步的一种协议。它的用途是把系统的时钟同步到 UTC 时区（世界协调时），其精度在局域网内可达到 0.1ms，在互联网上绝大多数情况其精度可以达到 1 ~ 50ms。

配置内容如下。

```
root@OpenWrt:~# cat /etc/config/system
config system
        option hostname OpenWrt
        option timezone UTC

config timeserver ntp
        list server     0.openwrt.pool.ntp.org
        list server     1.openwrt.pool.ntp.org
        list server     2.openwrt.pool.ntp.org
        list server     3.openwrt.pool.ntp.org
        option enabled 1
        option enable_server 0
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show system
system.@system[0]=system
system.@system[0].hostname=OpenWrt
system.@system[0].timezone=UTC
system.ntp=timeserver
system.ntp.server=0.openwrt.pool.ntp.org 1.openwrt.pool.ntp.org
2.openwrt.pool.ntp.org 3.openwrt.pool.ntp.org
```

```
system.ntp.enabled=1
system.ntp.enable_server=0
```

其中 system.@system 匿名节点选项的参数说明如表 11-1 所示。

表 11-1　system.@system 匿名节点选项参数说明

选项	说明	可选值及说明	必填
hostname	设备的主机名称，默认为 OpenWrt	字符串	否
buffersize	内核信息输出的尺寸，默认由内核指定	数字	否
conloglevel	控制平台日志记录的级别，级别越高，记录信息越多，默认为 7	1~8 数字	否
cronloglevel	CRON 服务写入系统 syslog 的日志级别，默认为 5	0：记录全部信息 8：记录命令被执行 9：只记录错误	否
klogconloglevel	内核向控制平台显示的信息级别，默认为 7	数字	否
log_file	syslog 日志文件路径，默认为 /var/log/messages	字符串，文件路径	否
log_ip	syslog 日志文件通过 IP 地址发送到另外一台机器上，默认为空	字符串，IP 地址	否
log_port	另外那台记录 syslog 机器的接收端口，默认为 514	数字，端口号	否
log_prefix	为通过网络发送的日志增加一个前缀，默认为空	字符串	否
log_size	日志文件的允许容量，默认为 16	数字，单位为 KB	否
timezone	时区，内容参考时区表，默认为 UTC	字符串，时区参数	否

Tips

时区表地址：http://wiki.openwrt.org/doc/uci/system#time_zones

system.ntp 选项的参数说明如表 11-2 所示。

表 11-2　system.ntp 选项参数说明

选项	说明	可选值及说明	必填
server	列表值，NTP 服务器的地址，默认值为空	字符串，列表值可以重复存在	否
enable	是否开启 NTP 功能	0：禁用 1：启用	否
enable_server	启动模式，客户端表示向远端获取，服务端表示向内网计算机提供 NTP 服务	0：启动客户端和服务端 1：启动服务端	否

举例，修改主机名称、时区、NTP 服务器地址（修改匿名节点需要使用 -X 先获得 ID，再根据 ID 修改）。

```
root@OpenWrt:~# uci show -X system
system.cfg02e48a=system
system.cfg02e48a.hostname=OpenWrt
system.cfg02e48a.timezone=UTC
system.ntp=timeserver
system.ntp.server=0.openwrt.pool.ntp.org 1.openwrt.pool.ntp.org
2.openwrt.pool.ntp.org 3.openwrt.pool.ntp.org
system.ntp.enabled=1
system.ntp.enable_server=0
root@OpenWrt:~# uci set system.cfg02e48a.hostname=AXWRT
root@OpenWrt:~# uci set system.cfg02e48a.timezone=CST-8
root@OpenWrt:~# uci set system.cfg02e48a.zonename=Asia/Shanghai
root@OpenWrt:~# uci delete system.ntp.server
root@OpenWrt:~# uci add_list system.ntp.server=ntp1.aliyun.com
root@OpenWrt:~# uci add_list system.ntp.server=ntp2.aliyun.com
root@OpenWrt:~# uci add_list system.ntp.server=ntp3.aliyun.com
```

保存确认修改。

```
root@OpenWrt:~# uci commit
root@OpenWrt:~# cat /etc/config/system

config system
        option hostname 'AXWRT'
        option timezone 'CST-8'
        option zonename 'Asia/Shanghai'

config timeserver 'ntp'
        option enabled '1'
        option enable_server '0'
        list server 'ntp1.aliyun.com'
        list server 'ntp2.aliyun.com'
        list server 'ntp3.aliyun.com'
```

重启，让设置生效。

```
root@OpenWrt:~# reboot
```

查看时间。

```
root@AXWRT:/# date
Wed Jan 28 11:38:57 CST 2015
```

11.2 服务管理

OpenWrt 系统比标准 x86 的 Linux 服务更简单一些，查看服务命令如下所示。

```
root@AXWRT:/# ls /etc/init.d
boot        done        led         network     sysfixtime   telnet
cron        dropbear    log         odhcpd      sysntpd      umount
dnsmasq     firewall    miniupnpd   sysctl      system
```

服务都可以发送几个标准命令，所有服务的启动和关闭方式都是相同的。

```

```

以后章节会用到以上这些语法，服务还有很多，这一章节不再做更多说明。

举例，关闭和重启网络服务。

```
root@AXWRT:/# /etc/init.d/network stop
root@AXWRT:/# /etc/init.d/network restart
```

11.3 SSH 远程登录

SSH（Secure Shell）是 Linux 系统中标配的远程命令管理协议，通过 SSH 即可访问系统的完整 Shell 权限，其作用类似我们进行的串口登录，但是串口登录一定要有线连接，SSH 可以通过网络直接访问，更方便我们进行调试。

由于防火墙限制 SSH 服务，所以 SSH 服务只面向 LAN 口（有线的或无线的都可以）开放，测试 SSH 要连入 LAN 口，如果一定要向 WAN 口开放也不是不可以，还记得防火墙那一章介绍的怎么创建一个 rule 规则吗？SSH 端口是 TCP 22，可以动手试试看。

11.3.1 SSH 配置

SSH 服务在 OpenWrt 下是通过一个名字叫 dropbear 的软件包实现的。

配置内容如下。

```
root@AXWRT:/# cat /etc/config/dropbear
config dropbear
        option PasswordAuth 'on'
```

```
        option RootPasswordAuth 'on'
        option Port         '22'
#       option BannerFile   '/etc/banner'
```

UCI 查看方法如下。

```
root@AXWRT:/# uci show dropbear
dropbear.@dropbear[0]=dropbear
dropbear.@dropbear[0].PasswordAuth=on
dropbear.@dropbear[0].RootPasswordAuth=on
dropbear.@dropbear[0].Port=22
```

dropbear.@dropbear 匿名节点选项的参数说明如表 11-3 所示。

表 11-3 dropbear.@dropbear 匿名节点选项参数说明

选项	说明	可选值及说明	必填
enable	默认是否开启，默认为 1	0：禁用 1：启用	否
PasswordAuth	登录时提示输入密码，默认为 1	0：禁用 1：启用	否
RootPasswordAuth	允许 root 使用密码登录，默认为 1	0：禁用 1：启用	否
RootLogin	允许 root 登录，默认为 1	0：禁用 1：启用	否
BannerFile	登录后显示特定的欢迎信息，默认为空	字符串，文件路径	否
Port	SSH 服务的端口号	数字，端口	否
SSHKeepAlive	开启服务端心跳，默认为 300	数字，单位为秒	否
IdleTimeout	空闲超时，默认为 0	数字，0 表示不开启，单位为秒	否

SSH 服务的默认状态已经足够好了，不需要修改，如果要修改需要重启 SSH 服务。

```
root@AXWRT:/# /etc/init.d/dropbear restart
```

11.3.2 SSH 登录

（1）系统默认的 root 用户没有密码，这个时候 SSH 是登录不上的，先使用 passwd 命令给 root 设置一个密码，密码要填写两次，屏幕不显示所填写内容。

```
root@AXWRT:/# passwd
Changing password for root
```

```
New password:
Retype password:
Password for root changed by root
```

（2）下载 SSH 客户端。PuTTY 是一个强大的工具，已经具备 SSH 功能了，只需要再打开一个 PuTTY 窗口，然后 Host Name 输入 MR808 开发板的 IP 地址（192.168.1.1），Port 输入端口号 22，Connection Type 选择 SSH，单击“Open”。SSH 客户端配置如图 11-1 所示。

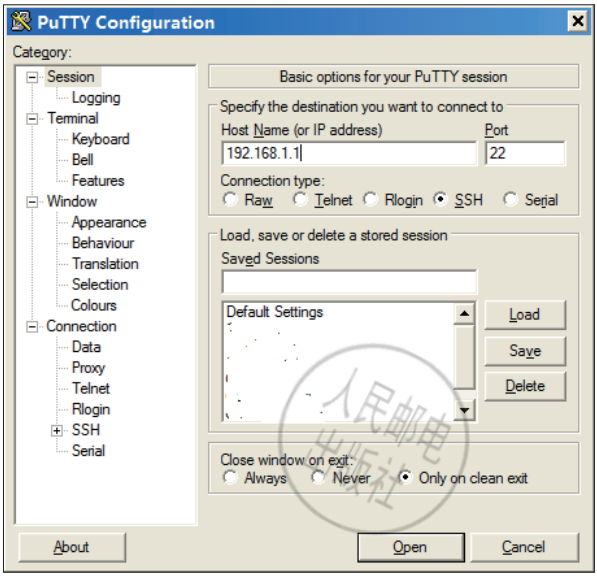


图 11-1　SSH 客户端配置

（3）第一次登录会提示警告信息，直接确认了就行，然后填写账户和密码，如下所示。

```
login as: root
root@192.168.1.1's password:


BusyBox v1.22.1 (2015-01-21 15:06:58 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

            ___    _  ___        ______  ______
           /   | | |/ / |      / / __ \/_  __/
          / /| | |   /| | /| / / /_/ / / /
         / ___ |/   | | |/ |/ / _, _/ / /
        /_/  |_/_/|_| |__/|__/_/ |_| /_/
 ---------------------------------------------------
 Axwrt is "smart-router Tutorial" customized based on
 Openwrt (Barrier Breaker, r43).
```

```
  * Opensource: www.geeqee.com/axwrt
  * Copyleft: Chengdu GeeQee Tech. Co.,Ltd.
  * Author: Sun Bing<hoowa.sun@gmail.com>
 -----------------------------------------------------
 Cooperative partners
  * www.ingchuang.com
  * www.geeqee.com
  * www.flamingoeda.com
  * www.wrtnode.com
 Words
  * Tutorial - http://www.ingchuang.com/feature/hoowa
  * MR808 - first tutorial's hardware.
  * WRTnode - MR808's core board.
 -----------------------------------------------------
root@AXWRT:~#
```

接下来你可以尽情地玩了，不用串口也没有关系啦。

11.4 SCP 文件复制

经过上面的配置，已经可以远程登录了，接下来我们要实现文件的复制，就是计算机与 SSH 之间如何进行远程文件的相互传输。当然也可以安装另外的 FTP 进行传输，但是太浪费了。直接用 SSH 附带的 SCP 协议就可以实现远程文件的传输。

11.4.1 安装 SCP 客户端

1 先下载 WinSCP，这是一个大名鼎鼎的 SCP 客户端软件。

Tips

WinSCP 官方网站：http://www.winscp.net

WinSCP 百度网盘下载地址：http://pan.baidu.com/s/1nv7hKBR

2 对下载包进行解压缩，然后双击图标进行“安装”，首先选择语言，单击“确认”，如图 11-2 所示。

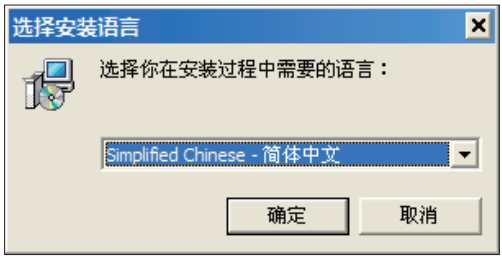
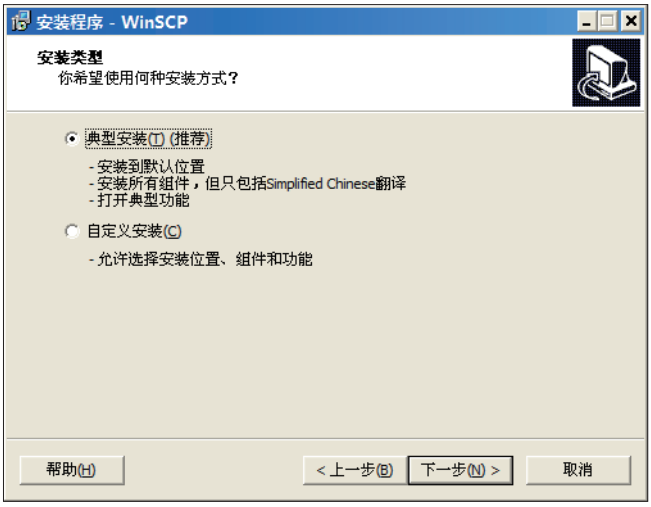


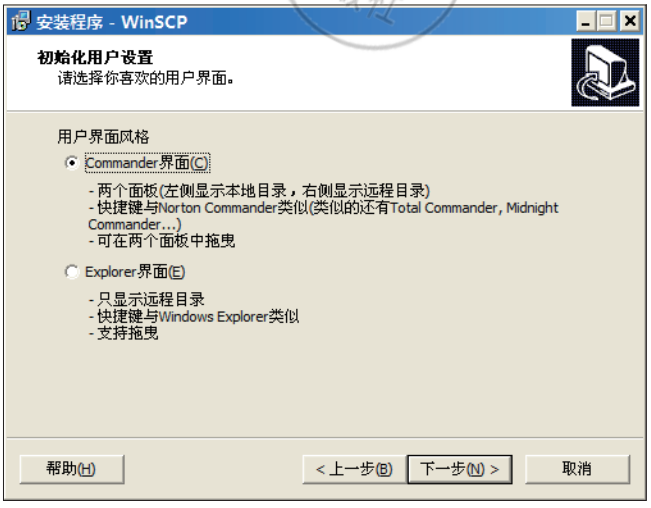
图 11-2 选择安装语言

3 单击“下一步”，选择同意许可协议，再单击“下一步”，安装类型选择“典型安装”，单击“下一步”，如图 11-3 所示。



■ 图 11-3 选择“典型安装”

4 用户界面选择“Commander 界面”，单击“下一步”，如图 11-4 所示。



■ 图 11-4 选择“Commander 界面”

5 单击“安装”，进入安装步骤，完成后单击“完成”。

11.4.2 SCP 操作

首次启动 WinSCP 会立即展现出登录创建窗口。WinSCP 配置界面如图 11-5 所示。

1. 填写登录信息。

（1）主机名：填写 MR808 开发板的 IP 地址，一般是 192.168.1.1。

（2）端口号：SSH 的端口号。

（3）用户名：填写 root。

（4）密码：填写 SSH 登录用的密码。

（5）文件协议：一定要选择 SCP，否则登录无法完成。

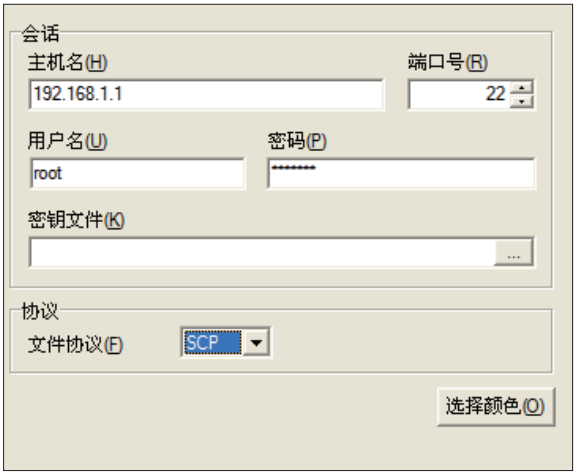


图 11-5 WinSCP 配置界面

2. 单击“保存”，会弹出新对话框，可以选上保存密码，单击“确认”，返回登录对话框。选中刚才创建的登录信息，单击对话框下角的“登录”。

3. 首次登录会提示一些警告信息，直接单击“是”，过程会报错找不到 groups 指令，跳过去即可。

4. 成功后，屏幕左侧为本机，屏幕右侧为 MR808 开发板系统。

5. 上传文件：在左侧选中文件拖动到右侧即可完成上传。

6. 下载文件：在右侧拖动文件到左侧即可完成下载。

7. 删除文件：在右侧选中文件，单击右键选“删除”即可。

第12章 存储器扩展

经过了前几章的学习，有些读者反映路由器的内存太小了，闪存也太小了，怎么办？本章将介绍如何扩展路由器存储器，包括存储器选型、初始化存储器和存储器扩容设置。本章内容是后续章节的基础，因此请读者认真阅读和操作。

12.1 实现原理

MR808 开发板具备两种存储器接口：一种是 USB 接口，另一种是 SATA 接口，这些接口可以足够满足我们扩展存储器的需要。配合 Linux 操作系统下的 swap 交换分区技术可以实现内存扩展（交换分区类似于 Windows 下的虚拟内存）。

在 OpenWrt 系统下扩展存储器容量，一般采用以下两种方式实现。

- 外部存储器替换 rootfs_data 分区的可写作用，也就是说替代现在的 /overlay，这样凡是被修改的文件都保存在外部磁盘。
- 直接将外部存储器挂载为 /mnt/extdisk 文件夹，这样就需要自己手工将需要的数据保存在里面。

正常启动 OpenWrt 涉及存储器的部分：

```
kernel → rootfs_rom → rootfs_data
```

启动了外部存储器的 OpenWrt：

```
kernel → rootfs_rom → rootfs_data → /dev/sda1
```

外部存储器在启动之前最后挂载的存储器是 rootfs_data，一旦启动后，在完成 rootfs_data 挂载后，还会再次挂载外部磁盘（通过读取 /etc/config/fstab 文件），从而完成替代 rootfs_data 的过程。

12.2 存储器选型

- **2.5 英寸笔记本电脑硬盘：**MR808 开发板支持一个 SATA 接口，可以直接连接笔记本电脑硬盘，通过 SATA 接口连接笔记本电脑硬盘，速度快，没有擦写次数限制，在极限测试下写速度可以在 MR808 开发板上跑到 25MB/s。

- **USB 移动硬盘或 U 盘**：性能其次，另外要注意供电稳定性，如果开发板还要再用到其他一些 USB 设备，可能会因为供电不足而导致异常。如果真的要用 USB 连接，本书建议使用 U 盘或 SD 卡方式，或者使用带独立供电的移动硬盘。

关于 Flash 闪存不得不说的故事：固态硬盘、U 盘、TF 卡都是基于 Flash 闪存技术的，这个技术很流行，但作为存储器，其有自身技术的局限性，主要来源于存储单元技术。

- **SLC**：Single Layer Cell（单层单元），SLC 的特点是成本高、容量小、速度快，每个存储单元有 1bit 数据，每个单元可擦写次数可达到 10 万次，一般使用在工业级产品和服务器级的 SSD 中，如果你的设备主要提供频繁擦写，使用 SLC 比较适合，但是其价格也要贵很多。
- **MLC**： Multi-Level Cell（多层单元），在相同的单元密度下可存储 2bit 数据，这是目前市面上消费级 SSD 使用的最主要存储技术，但是由于密度增加和算法原因，每个单元理论可擦写次数为 3000~10000 次。
- **TLC**: Triple-level cell（多倍单元），在相同的单元密度下可存储 3bit 数据，这是目前低价位 U 盘、TF 卡常用的，其特点是密度大、成本低，缺点是速度慢、容易损坏，每个单元理论可擦写次数为 500~1000 次。

因此，本书推荐，如果仅仅是学习使用，可以用个 U 盘简单地扩大容量。如果想要做存储器扩展，最好使用 MR808 开发板自带的 SATA 口连接一块硬盘。

12.3 初始化存储器

在初始化存储器之前，请先关闭 MR808 开发板，插入笔记本电脑硬盘，并且用螺丝固定在 MR808 开发板上。如果没有硬盘，可以用一个 U 盘替代，然后再开机。

12.3.1 存储器软件包

保持 MR808 开发板外网连接正常，然后安装所需的软件包。

```
root@SmartRouter:/# opkg update
root@SmartRouter:/# opkg install kmod-usb-core kmod-usb-ohci kmod-usb-
uhci kmod-usb2
root@SmartRouter:/# opkg install kmod-usb-storage kmod-usb-storage-
extras kmod-fs-ext4
root@SmartRouter:/# opkg install kmod-fs-vfat kmod-nls-utf8 kmod-nls-
cp437
root@SmartRouter:/# opkg install usbutils fdisk block-mount e2fsprogs
blkid
```

安装过程中会提示很多信息，也会有警告，安装好之后执行重启就会正常。

```
root@OpenWrt:~# reboot
```

12.3.2 识别存储器

查看系统识别到的磁盘，命令如下所示。

```
root@OpenWrt:~# fdisk -l

.......

Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000d3889
```

如上所示，本书使用一个 320GB 的笔记本电脑硬盘做演示，系统中已经可以看到该硬盘被识别出来，并且赋予了**设备节点** /dev/sda。设备节点跟随插入 USB 口的设备顺序变化而产生变化。

在 MR808 开发板上，设备号节点是以 /dev/sd[X][N] 方式来编码的，其中 X 表示设备序列号，N 表示分区编号。

- /dev/sda：第一个插入的存储器。
- /dev/sda1：第一个存储器的第一个分区。
- /dev/sdb：第二个存储器。
- 等等……

可能出现的问题如下。

（1）如果设备重启，顺序可能会产生变化，/dev/sda 可能会变成 /dev/sdb，我们将介绍如何解决。

（2）后面的流程不论是硬盘、U 盘都是相同的，唯一不同的就是设备节点，每次检查设备节点都可以使用 fdisk -l 指令。

12.3.3 存储器分区

分区技术上我们使用 fdisk 命令，本书计划分两个分区。

- 分区一：容量 1GB，用来做交换区扩充内存，这样 U 盘就不太适合了。
- 分区二：剩余全部容量，用来做存储器扩展。

进入分区模式。

```
root@OpenWrt:~# fdisk /dev/sda

Welcome to fdisk (util-linux 2.24.1).
Changes will remain in memory only, until you decide to write them.
```

```
Be careful before using the write command.
```

输入 d 指令删除掉旧分区（如果你有多个分区，删除程序会让你输入删除哪一个，范围是 1~4，如果只有一个分区，会默认被删除掉）。

```
Command (m for help): d
No partition is defined yet!
```

用 n 指令开始创建分区。

```
Command (m for help): n
```

用 p 指令选择被创建分区类型（创建主分区）。

```
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
```

再输入分区编号（主分区编号为 1~4）。

```
Partition number (1-4, default 1): 1
```

设置被创建的分区尺寸，First sector 不用填写，在 Last sector 填写“+1G”表示创建 1GB 大小的分区。

```
First sector (2048-625142447, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-625142447, default
625142447): +1G

Created a new partition 1 of type 'Linux' and of size 1 GiB.
```

用 t 指令将所创建分区设置为 swap 交换内存类型，然后输入编号 82。

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'.
```

再用全部剩余容量创建第二个分区，语法与创建第一个分区相同。

```
Command (m for help): n

Partition type:
   p   primary (1 primary, 0 extended, 3 free)
```

```
   e   extended
Select (default p): p
Partition number (2-4, default 2):
First sector (2099200-625142447, default 2099200):
Last sector, +sectors or +size{K,M,G,T,P} (2099200-625142447, default
625142447):

Created a new partition 2 of type 'Linux' and of size 297.1 GiB.
```

检查创建列表。

```
Command (m for help): p
Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000d3889

Device     Boot     Start       End    Blocks  Id System
/dev/sda1           2048   2099199   1048576  82 Linux swap / Solaris
/dev/sda2        2099200 625142447 311521624  83 Linux
```

用 w 指令将创建保存起来。

```
Command (m for help): w

The partition table has been altered.
Calling ioctl() to re-read partition tabl[ 1374.640000]  sda: sda1 sda2
e.
Syncing disks.
```

12.3.4 存储器格式化

看看两个分区的设备节点吧，/dev/sda1 是交换区，/dev/sda2 是存储区。

```
root@OpenWrt:~# fdisk -l

......

Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000d3889

Device     Boot     Start       End    Blocks  Id System
```

```
/dev/sda1            2048    2099199    1048576  82 Linux swap / Solaris
/dev/sda2         2099200 625142447 311521624  83 Linux
```

格式化交换分区。

```
root@OpenWrt:~# mkswap /dev/sda1
Setting up swapspace version 1, size = 1073737728 bytes
```

采用 EXT4 分区类型格式化数据区。

```
root@OpenWrt:~# mkfs.ext4 /dev/sda2
mke2fs 1.42.4 (12-June-2012)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
19472384 inodes, 77880406 blocks
3894020 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=0
2377 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
2654208,4096000, 7962624, 11239424, 20480000, 23887872, 71663616

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information:done
```

12.4 存储器扩容设置

挂载的作用是将磁盘连接到特定文件夹上，这样就可以对磁盘进行读写操作了。

在处理挂载之前，请先重启一下设备。

```
root@OpenWrt:~# reboot
```

12.4.1 rootfs_data 数据迁移

核对当前存储器的设备节点，防止节点有变化，误操作。

```
root@OpenWrt:/# fdisk -l
```

创建临时文件夹。

```
root@OpenWrt:~# mkdir /mnt/extdisk
```

将 /dev/sda2 挂载到临时文件夹。

```
root@OpenWrt:~# mount /dev/sda2 /mnt/extdisk
[  115.750000] EXT4-fs (sda2): couldn't mount as ext3 due to feature
incompatibilities
[  115.750000] EXT4-fs (sda2): couldn't mount as ext2 due to feature
incompatibilities
[  115.950000] EXT4-fs (sda2): mounted filesystem with ordered data
mode. Opts: (null)
```

复制整个 /overlay 中的内容到存储器。

```
root@OpenWrt:~# tar -C /overlay -cvf - . | tar -C /mnt/extdisk -xf -
```

刷新写入缓存。

```
root@OpenWrt:~# sync
```

卸载临时挂载，完成这部分流程。

```
root@OpenWrt:~# umount /mnt/extdisk
```


12.4.2 获得存储器 UUID

前面说过，磁盘挂载可能存在节点变化，如果要实现自动挂载需要 UUID 技术。UUID 是对设备的一个描述，所有的 UUID 节点都是唯一的，这样不论设备节点如何变化，始终都可以找到所对应的磁盘分区。

获得存储器的扩展分区 UUID（演示中是本书设备的 UUID）。

```
root@OpenWrt:~# blkid /dev/sda2
/dev/sda2: UUID="61299cb7-035e-4a3c-b686-1257342c8867" TYPE="ext4"
PARTUUID="000d3889-02"
```

12.4.3 自动挂载配置

配置内容如下。

```
root@OpenWrt:~# cat /etc/config/fstab
config 'global'
        option  anon_swap       '0'
```

```
        option  anon_mount      '0'
        option  auto_swap       '1'
        option  auto_mount      '1'
        option  delay_root      '5'
        option  check_fs        '0'

config 'mount'
        option  target  '/mnt/sda1'
        option  uuid    '3cefc1eb-c9c9-4fc6-ba80-ff2714a6c2aa'
        option  enabled '0'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show fstab
fstab.@global[0]=global
fstab.@global[0].anon_swap=0
fstab.@global[0].anon_mount=0
fstab.@global[0].auto_swap=1
fstab.@global[0].auto_mount=1
fstab.@global[0].delay_root=5
fstab.@global[0].check_fs=0
fstab.@mount[0]=mount
fstab.@mount[0].target=/mnt/sda1
fstab.@mount[0].uuid=3cefc1eb-c9c9-4fc6-ba80-ff2714a6c2aa
fstab.@mount[0].enabled=0
```

fstab.@global 匿名节点选项的参数说明如表 12-1 所示。

表 12-1 fstab.@global 匿名节点选项参数说明

选项	说明	可选值及说明	必填
anon_swap	自动挂载任何一个外置磁盘上的交换区，默认为 0	0：禁用 1：启用	否
anon_mount	当 hotplug 程序执行到挂载的时候，自动挂载没有配置的存储器分区到 /mnt/ 下，默认为 0	0：禁用 1：启用	否
auto_swap	启用自动挂载 swap 的设置，默认为 1	0：禁用 1：启用	否
auto_mount	启动自动挂载存储器分区，默认为 1	0：禁用 1：启用	否
delay_root	挂载之前等待多少秒，等待存储器连接上线，默认为 5	数字，单位为秒	否
check_fs	在挂载时对 ext 类型的分区进行检测，默认为 0	0：禁用 1：启用	否

fstab.@mount 匿名节点选项的参数说明如表 12-2 所示。

表 12-2　fstab.@mount 匿名节点选项参数说明

选项	说明	可选值及说明	必填
target	挂载到什么路径，默认为空	字符串，路径	是
device	被挂载的设备符号，如果有 UUID 可以不填此参数，默认为空	字符串，设备符号	否
uuid	被挂载的设备 UUID，默认为空	字符串，设备 UUID 编码	否
fstype	被挂载的分区类型，默认为 auto	auto：自动 ext4、ext3、fat32	否
options	挂载的命令参数，默认为空	字符串	否
enabled	启用自动挂载，默认为 1	0：禁用 1：启用	否
enabled_fsck	启用在挂载的时候对分区进行文件系统检测，默认为 0	0：禁用 1：启用	否

12.4.4　存储区自动挂载

存储区的设备节点是 /dev/sda2，但是现在我们要使用 UUID 来实现挂载。

匿名节点首先查看 fstab.@mount[0] 节点的 ID 号。

```
root@OpenWrt:/# uci show -X fstab
fstab.cfg023fd6=global
fstab.cfg023fd6.anon_mount=0
fstab.cfg023fd6.auto_swap=1
fstab.cfg023fd6.auto_mount=1
fstab.cfg023fd6.delay_root=5
fstab.cfg023fd6.check_fs=0
fstab.cfg023fd6.anon_swap=1
fstab.cfg044d78=mount
fstab.cfg044d78.target=/mnt/sda1
fstab.cfg044d78.uuid=3cefc1eb-c9c9-4fc6-ba80-ff2714a6c2aa
fstab.cfg044d78.enabled=0
```

获取存储分区的 UUID 并修改。

```
root@OpenWrt:/# blkid /dev/sda2
/dev/sda2: UUID="61299cb7-035e-4a3c-b686-1257342c8867" TYPE="ext4"
PARTUUID="000d3889-02"
root@OpenWrt:/# uci set fstab.cfg044d78.uuid=61299cb7-035e-4a3c-b686-
```

```
1257342c8867
```

配置挂载参数。

```
root@OpenWrt:/# uci set fstab.cfg044d78.target=/overlay
root@OpenWrt:/# uci set fstab.cfg044d78.fstype=ext4
root@OpenWrt:/# uci set fstab.cfg044d78.options='rw,sync'
root@OpenWrt:/# uci set fstab.cfg044d78.enabled=1
root@OpenWrt:/# uci set fstab.cfg044d78.enabled_fsck=0
```

保存配置。

```
root@OpenWrt:/# uci commit fstab
```

重新启动使设置生效。

```
root@OpenWrt:~# reboot
```

检测是否挂载成功。

```
root@OpenWrt:~# mount|grep sda2
/dev/sda2 on /overlay type ext4 (rw,sync,relatime,user_
xattr,barrier=1,data=ordered)
```

12.4.5 swap 交换区自动挂载

存储区挂载完成后，/etc/config/fstab 这个文件就是存储器上的版本，也就是说看起来好像没修改过。那就对了，挂载存储区时 /overlay 是 rootfs_data，现在是在外部存储器上了，这个文件是在之前复制的，所以没变化，不过不会有影响。我们现在就用这个文件实现交换区的挂载。

匿名节点首先查看节点的 ID 号。

```
root@OpenWrt:/# uci show -X fstab
fstab.cfg023fd6=global
fstab.cfg023fd6.anon_mount=0
fstab.cfg023fd6.auto_swap=1
fstab.cfg023fd6.auto_mount=1
fstab.cfg023fd6.delay_root=5
fstab.cfg023fd6.check_fs=0
fstab.cfg023fd6.anon_swap=1
fstab.cfg044d78=mount
fstab.cfg044d78.target=/mnt/sda1
fstab.cfg044d78.uuid=3cefc1eb-c9c9-4fc6-ba80-ff2714a6c2aa
fstab.cfg044d78.enabled=0
```

利用找到的节点 ID 配置打开匿名交换区挂载。

```
root@OpenWrt:~# uci set fstab.cfg023fd6.anon_swap=1
root@OpenWrt:~# uci set fstab.cfg023fd6.auto_swap=1
root@OpenWrt:~# uci commit fstab
```

重启使设置生效。

```
root@OpenWrt:~# reboot
```

检测交换区是否完成挂载（显示容量大约 1GB 表示交换区挂载成功）。

```
root@OpenWrt:/# cat /proc/meminfo |grep Swap
SwapCached:            0 kB
SwapTotal:       1048572 kB
SwapFree:        1048572 kB
```

12.4.6 关于挂载的注意事项

（1）系统已经改变：从本章开始，以后所做的任何修改都将会写在你的磁盘上了，再也不会变化 Flash 了，当然存储器扩展后容量也大得惊人。

（2）去掉扩展的存储器会如何：如果拔掉存储器，系统会默认返回使用 Flash 上的 rootfs_data 来实现 /overlay，启动可以正常完成，但是之后安装的东西都不见了，系统回到原点。不过别怕，再插上之前的那个磁盘，又可以恢复。除非那个外面的存储器坏掉了，那时就要重复本章前面的内容重建了。

（3）插入其他设备会报错：可能是供电异常或磁盘顺序变化，建议重启再看看。

（4）检查 /etc/config/fstab 配置文件：当系统启动后，我们检查这个文件发现怎么还是旧的？对了，因为你在挂载之前修改的是 rootfs_data 下的 /etc/config/fstab，而挂载之后看到的 /etc/config/fstab 是磁盘上的，磁盘上这个还没改过呢！不过没关系，不影响使用。

第13章 网络存储与文件共享

本章介绍网络存储中使用较多的两种文件共享方式，分别是 Windows 网上邻居和 FTP，通过 Windows 网上邻居和 FTP 实现文件共享，就可以把需要存储的东西保存在 MR808 开发板的硬盘里了。

文件共享和传输的协议有很多种，Windows 文件共享服务器更适合局域网内部文件共享管理，这种方式简单容易、实用方便。FTP 文件服务器适合局域网及公网远程文件共享服务。

13.1 Windows 文件共享实现

13.1.1 创建文件夹 /pub

```
root@OpenWrt:~# mkdir /pub
```

给予 /pub 任何进程都可以读写的权限，这样在以后的演示中不会再遇到权限问题。

```
root@OpenWrt:~# chmod 777 /pub
```

实现 Windows 与 Linux 之间共享数据的软件叫 Samba，该软件可以在 Linux 下实现网上邻居，并且可以被 Windows 探测发现。

13.1.2 安装 Samba

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install samba36-server
```

13.1.3 Samba 配置说明

/etc/config/samba 配置内容如下。

```
config samba
        option 'name'                'OpenWrt'
        option 'workgroup'            'WORKGROUP'
        option 'description'            'OpenWrt'
```

```
        option 'homes'                '1'

config 'sambashare'
        option 'name'                   'shares'
        option 'path'                   '/mnt/extdisk'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show samba
samba.@samba[0]=samba
samba.@samba[0].name=OpenWrt
samba.@samba[0].workgroup=WORKGROUP
samba.@samba[0].description=OpenWrt
samba.@samba[0].homes=1
samba.@sambashare[0].name=shares
samba.@sambashare[0].path=/mnt/extdisk
```

- samba.@samba 匿名节点只能有一个，是通用配置，samba.@samba 匿名节点选项参数说明如表 13-1 所示。
- samba.@sambashare 匿名节点允许有多个，每个都代表一个共享的文件夹，samba.@sambashare 匿名节点选项参数说明如表 13-2 所示。

表 13-1　samba.@samba 匿名节点选项参数说明

选项	说明	可选值及说明	必填
charset	文件名编码格式，默认为 UTF-8	UTF-8、cp936	否
name	服务的名称，默认为主机名称	字符串	否
workgroup	网上邻居工作组，默认为 WORKGROUP	字符串	否
description	对设备的描述信息	字符串	否
homes	是否启动用户文件夹共享模式，默认为 0	0：禁用 1：启用	否

表 13-2　samba.@sambashare 匿名节点选项参数说明

选项	说明	可选值及说明	必填
name	共享的名字，默认为空	字符串	是
path	共享所对应的设备上的路径，默认为空	字符串，共享路径	是
users	指定什么用户可以访问这个共享，默认为空	字符串，有权限的系统用户	否

续表

选项	说明	可选值及说明	必填
read_only	共享是否为只读模式，默认为 no	no：否定 yes：肯定	否
guest_ok	是否支持访客模式，默认为 no	no：否定 yes：肯定	否
create_mask	由共享所创建的文件权限，默认为 0744	数字，权限编码 4 位	否
dir_mask	由共享所创建的目录权限，默认为 0755	数字，权限编码 4 位	否

13.1.4 配置共享目录

创建一个新的匿名节点，创建成功后将得到这个节点的 ID 号。

```
root@OpenWrt:~# uci add samba sambashare
cfg05e23c
```

增加一个 sambashare 共享 /pub 路径到 Windows 共享中。

```
root@OpenWrt:~# uci set samba.cfg05e23c.name='shares'
root@OpenWrt:~# uci set samba.cfg05e23c.path='/pub'
root@OpenWrt:~# uci set samba.cfg05e23c.guest_ok='yes'
root@OpenWrt:~# uci set samba.cfg05e23c.create_mask='0777'
root@OpenWrt:~# uci set samba.cfg05e23c.dir_mask='0777'
root@OpenWrt:~# uci set samba.cfg05e23c.read_only='no'
root@OpenWrt:~# uci commit samba
```

启动 samba 服务。

```
root@OpenWrt:~# /etc/init.d/samba start
```

设置 samba 服务自动启动。

```
root@OpenWrt:~# /etc/init.d/samba enable
```

13.1.5 Windows 连接方法

1 在开始菜单的“运行”中输入 MR808 开发板的 IP 地址（只能在 LAN 口访问），然后按回车键会进行连接，成功后会弹出 MR808 开发板的共享目录。开始菜单运行界面如图 13-1 所示，MR808 开发板的共享目录界面如图 13-2 所示。

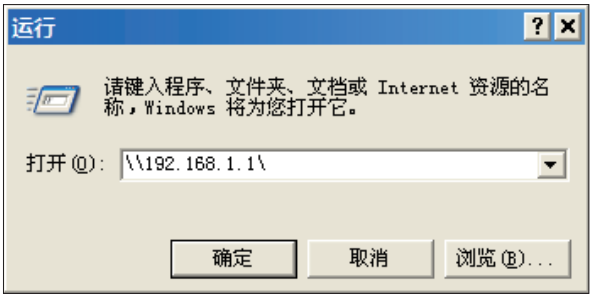


图 13-1　开始菜单运行界面

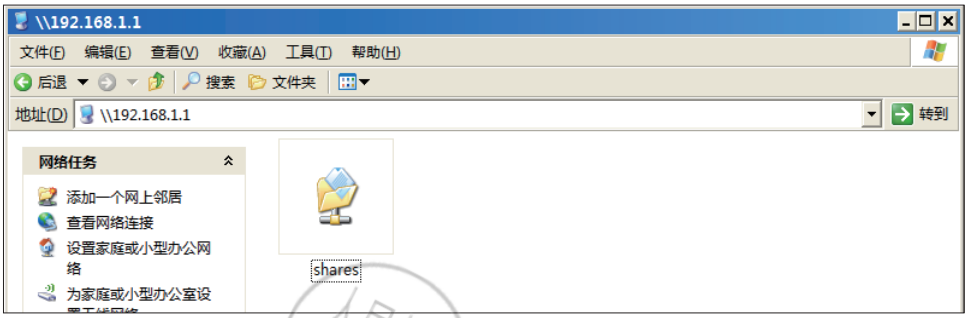


图 13-2　MR808 开发板共享目录

2 单击进入 shares，就可以进入 MR808 开发板的共享磁盘，像操作本地分区一样对它进行操作，MR808 开发板共享目录的内容如图 13-3 所示。

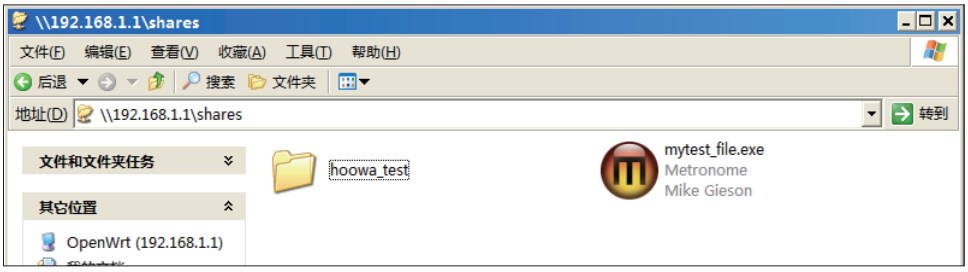


图 13-3　MR808 开发板共享目录内容

3 查看 MR808 开发板中的内容。

```
root@OpenWrt:~# ls -l /pub/
drwxrwxrwx    2 nobody   nogroup       4096 Jan 28 12:56 hoowa_test
-rwxrw-rw-    1 nobody   nogroup    3760635 Nov 27  2010 mytest_file.exe
```

13.2 FTP 文件共享实现

有时候，可能你需要远程共享服务并且可以支持不同的设备，使用 FTP 是最简单方便的方式。在这里我们使用 Pure-FTPD，它是一个快速、高效、轻便、安全的 FTP 服务器，非常适合在 OpenWrt 下使用。

13.2.1 安装 Pure-FTPD

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install pure-ftpd
```

13.2.2 Pure-FTPD 配置说明

/etc/config/pure-ftpd 配置内容如下。

```
root@OpenWrt:~# cat /etc/config/pure-ftpd
config pure-ftpd
        option port             '21'
        option noanonymous      '1'
        option chrooteveryone   '1'
        option maxclientsperip  '10'
        option maxclientsnumber '4'
        option peruserlimits    '3:4'
        option umask            '133:022'
        option authentication   'unix'
        option enabled          '0''
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show pure-ftpd
pure-ftpd.@pure-ftpd[0]=pure-ftpd
pure-ftpd.@pure-ftpd[0].port=21
pure-ftpd.@pure-ftpd[0].noanonymous=1
pure-ftpd.@pure-ftpd[0].chrooteveryone=1
pure-ftpd.@pure-ftpd[0].maxclientsperip=10
pure-ftpd.@pure-ftpd[0].maxclientsnumber=4
pure-ftpd.@pure-ftpd[0].peruserlimits=3:4
pure-ftpd.@pure-ftpd[0].umask=133:022
pure-ftpd.@pure-ftpd[0].authentication=unix
pure-ftpd.@pure-ftpd[0].enabled=0
```

pure-ftpd.@pure-ftpd 匿名节点选项的参数说明如表 13-3 所示。

表 13-3　pure-ftpd.@pure-ftpd 匿名节点选项参数说明

选项	说明	可选值及说明	必填
port	FTP 服务的端口，默认为 21	数字，端口	是
noanonymous	禁止匿名 FTP 访问	0：禁用 1：启用	否
chrooteveryone	允许 FTP 使用者访问整个服务器，默认为 0	0：禁用 1：启用	否
maxclientsperip	同一 IP 最大允许客户端数量，默认为 10	数字	否
maxclientsnumber	最大允许客户端数量，默认为 4	数字	否
peruserlimits	单一用户最大量限制，默认为 3:4	字符串，格式为“用户数：匿名用户数”，如 3:4	否
umask	创建文件和文件夹的权限，默认为 133:022	字符串，格式为“文件数：文件夹数”，如 000:000 表示完整权限	否
authentication	账号验证方法，默认为 puredb: /etc/pureftpd.pdb	字符串	否
enabled	启动 Pure-FTPD 服务	0：禁用 1：启用	否
fortunesfile	欢迎词文件路径，默认为空	字符串，路径	否
dontresolve	不进行解析，提高连接速度	0：禁用 1：启用	否
passiveportrange	FTP 被动模式端口范围	字符串，格式为“开始端口 – 结束端口”，如 65500–65535	否

13.2.3　配置 Pure-FTPD

默认的 Pure-FTPD 配置已经可以正常使用了，不用全部修改，修改部分即可。

取得匿名节点的 ID。

```
root@OpenWrt:~# uci show -X pure-ftpd
pure-ftpd.cfg0294fc=pure-ftpd
pure-ftpd.cfg0294fc.port=21
pure-ftpd.cfg0294fc.noanonymous=1
pure-ftpd.cfg0294fc.chrooteveryone=1
pure-ftpd.cfg0294fc.maxclientsperip=10
pure-ftpd.cfg0294fc.maxclientsnumber=4
pure-ftpd.cfg0294fc.peruserlimits=3:4
```

```
pure-ftpd.cfg0294fc.umask=133:022
pure-ftpd.cfg0294fc.authentication=unix
pure-ftpd.cfg0294fc.enabled=0
```

设置文件和目录的创建权限为 000:000，即全部权限。

```
root@OpenWrt:~# uci set pure-ftpd.cfg0294fc.umask='000:000'
```

设置 Pure-FTPD 可以启动。

```
root@OpenWrt:~# uci set pure-ftpd.cfg0294fc.enabled=1
```

设置关闭解析，提高速度。

```
root@OpenWrt:~# uci set pure-ftpd.cfg0294fc.dontresolve=1
```

设置被动模式端口范围。

```
root@OpenWrt:~# uci set pure-ftpd.cfg0294fc.passiveportrange='50000:51000'
```

设置验证方式为 puredb。

```
root@OpenWrt:~# uci set pure-ftpd.cfg0294fc.authentication='puredb:/
etc/pureftpd.pdb'
```

保存。

```
root@OpenWrt:~# uci commit pure-ftpd
```

启动 FTP 服务。

```
root@OpenWrt:~# /etc/init.d/pure-ftpd start
```

设置 FTP 服务自动启动。

```
root@OpenWrt:~# /etc/init.d/pure-ftpd enable
```

13.2.4 创建 FTP 账户

上一步我们已经制定了验证用的账户文件 /etc/pureftpd.pdb，这个文件是由 pure-pw 命令生成的。可以通过该命令产生 FTP 账户。

pure-pw 语法格式如下。

```
增加 FTP 新账户：pure-pw useradd [ 账户名称 ] -u [ 对应系统账户 ] -d [FTP 服务器文件
路径 ]
删除 FTP 账户：pure-pw userdel [ 账户名称 ]
修改 FTP 账户密码：pure-pw passwd [ 账户名称 ]
列出全部 FTP 账户：pure-pw list
生效数据：pure-pw mkdb
```

创建 /pub/hoowa 文件夹，并且所有人为 nobody，所有组为 nogroup。

```
root@OpenWrt:~# mkdir -p /pub/hoowa
root@OpenWrt:~# chown nobody:nogroup /pub/hoowa
```

为了保证与 Samba 一致，所创建的 FTP 账号请对应到系统账户 nobody 上。填写密码时所填写的密码不显示出来，重复填写两次密码，最后创建成功。

```
root@OpenWrt:~# pure-pw useradd hoowa -u nobody -d /pub/hoowa
Password:
Enter it again:
```

查看 FTP 所有账号。

```
root@OpenWrt:~# pure-pw list
hoowa                /pub/hoowa/./
```


生效数据文件。

```
root@OpenWrt:~# pure-pw mkdb
```

13.2.5 FTP 穿越防火墙

如果不需要外网访问 FTP 服务，即可跳过这一节。默认情况，防火墙中开启模式，WAN 口是不能对外开放 FTP 服务的。如果要允许 WAN 口直接访问系统的 FTP 服务，需要设置防火墙 rule 规则。

在以上配置中，FTP 使用 21 号端口，另外被动模式（目前使用最广泛的 FTP 传输模式）使用了 50000~51000 范围的端口，因此我们要在防火墙规则上允许 WAN 口的访问进入。

向防火墙增加匿名 rule 节点，记录下节点 ID。

```
root@OpenWrt:~# uci add firewall rule
cfg1692bd
```

利用节点 ID 进行规则设置。

```
root@OpenWrt:~# uci set firewall.cfg1692bd.name='wan-ftp'
```

```
root@OpenWrt:~# uci set firewall.cfg1692bd.src='wan'
root@OpenWrt:~# uci set firewall.cfg1692bd.dest_port='21 50000-51000'
root@OpenWrt:~# uci set firewall.cfg1692bd.target='ACCEPT'
root@OpenWrt:~# uci set firewall.cfg1692bd.proto='tcp'
```

保存防火墙配置，然后重置防火墙。

```
root@OpenWrt:~# uci commit firewall
root@OpenWrt:~# /etc/init.d/firewall reload
```

防火墙设置完毕后，就可以通过 WAN 口登录 FTP 服务了，LAN 口默认是允许登录的。

13.2.6 WinSCP 做 FTP 客户端

WinSCP 是一个强大的客户端软件，它支持 SFTP、FTP、SCP 三种模式，这里我们仅使用 FTP 功能进行测试连接。

1 在 WinSCP 登录界面单击“新建”，在新窗口中填写 FTP 连接信息。主机名填写 MR808 开发板 IP 地址，端口填写 21，用户名与密码填写刚才创建的 FTP 账号与密码，文件协议选择 FTP。WinSCP 登录配置界面如图 13-4 所示。

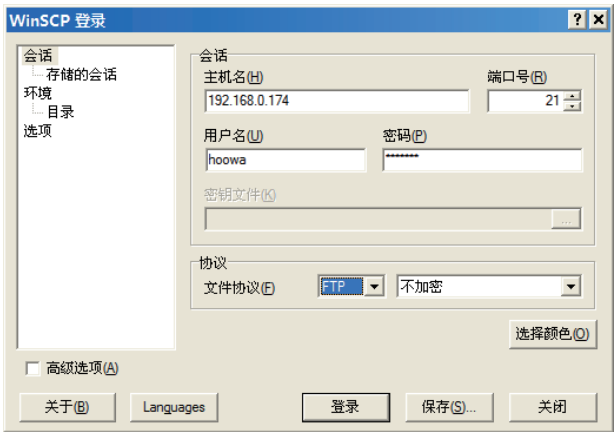


图 13-4 WinSCP 登录配置界面

2 单击“保存”，在新对话框选择“保存密码”，单击“确定”，返回登录对话框。单击刚才创建的连接信息，单击对话框下角的“登录”。

3 登录后将显示出文件传输界面，界面操作与 SCP 相同，由于启动了限制，所以登录账号只能看到自己的文件夹，相对更安全。文件传输界面如图 13-5 所示。

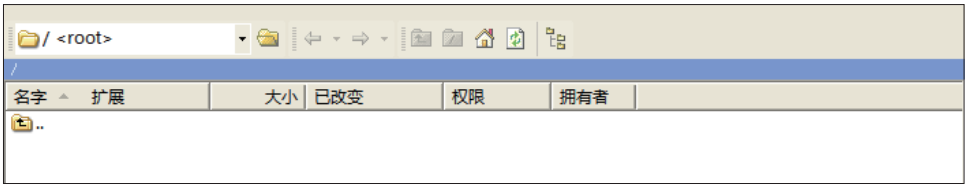


图 13-5　文件传输界面

13.2.7　浏览器作 FTP 客户端

其实，任何一个浏览器都可以作为 FTP 客户端，当然只能下载，不能上传，路径写法如下所示。

```
ftp:// 地址 : 端口
```

例如，在浏览器上填写如下内容。

```
ftp://192.168.1.1:21
```

屏幕中将弹出对话框，填写上账号和密码即可完成登录，浏览器登录界面如图 13-6 所示。

图 13-6　浏览器登录界面

第14章 智能路由器媒体中心

“忽闻水上琵琶声，主人忘归客不发。转轴拨弦三两声，未成曲调先有情。”

IT 男们为什么不为你的女朋友准备一台网络音箱呢？在她结束繁忙工作，下班回到家后，可以一边开心地听音乐，一边给你烧菜。本章将介绍智能路由器对 USB 声卡的支持、如何使用安卓手机来控制 MR808 开发板播放音乐、DLNA 数字媒体服务器和使用苹果 AirPlay 播放音乐。

14.1 USB 声卡驱动支持

14.1.1 媒体播放必备条件

（ 1 ）具备 USB 总线接口类型的声卡。

（ 2 ）声卡支持 ALSA 或 OSS 驱动框架。

（ 3 ）OpenWrt 系统中具备了大量的多媒体库可以播放声音。

14.1.2 选择合适的 USB 声卡

支持标准驱动的 USB 声卡芯片非常多，价格也很便宜。在购买 USB 声卡的时候不能直接选择是否支持标准驱动程序，并且卖 USB 声卡的经销商们也不懂得什么芯片、什么标准驱动程序，因此本书给大家几个小小的建议。

USB 音箱：如果你的房间不是很大，或并不追求 Hi-Fi 音质，那么市面上有很多带 USB 声卡的音箱，价格很便宜，在十几元钱到 100 元钱左右，这部分音箱一般是支持标准驱动程序的。不过也要注意供电问题，如果音箱功率比较高，并且没有独立供电，恐怕 MR808 开发板的供电带不动。

USB 声卡：一般价格比较便宜的都可以选择，购买时可以问一下卖家声卡是否支持 Windows XP 即插即用，无需驱动程序，这一类的声卡都可以很好地和 Linux 下标准驱动框架兼容。

14.1.3 安装声卡驱动

确保 MR808 开发板已经正常连接外网，安装驱动程序和软件包。

```
root@OpenWrt:~# opkg update
```

```
root@OpenWrt:~# opkg install kmod-usb-audio
root@OpenWrt:~# opkg install alsa-utils
```

查看没插入声卡情况下的 USB 设备。

```
root@OpenWrt:~# lsusb
Bus 001 Device 002: ID 0409:005a NEC Corp. HighSpeed Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 003: ID 152d:2329 JMicron Technology Corp. / JMicron USA
Technology Corp. JM20329 SATA Bridge
```

插入声卡设备，系统将提示如下信息。

```
[  184.330000] usb 1-1.2: new full-speed USB device number 4 using
ehci-platform
[  184.470000] usb 1-1.2: no of_node; not parsing pinctrl DT
[  189.430000] usb 1-1.2: USB disconnect, device number 4
[  189.730000] usb 1-1.2: new full-speed USB device number 5 using
ehci-platform
[  189.870000] usb 1-1.2: no of_node; not parsing pinctrl DT
```

测试声卡插入后的 USB 设备，确认声卡插入正确（本系统多了一个 biforst 设备）。

```
root@OpenWrt:~# lsusb
Bus 001 Device 002: ID 0409:005a NEC Corp. HighSpeed Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 003: ID 152d:2329 JMicron Technology Corp. / JMicron USA
Technology Corp. JM20329 SATA Bridge
Bus 001 Device 005: ID 19a8:2036 Biforst Technology Inc.
```

重新启动系统，实现加载驱动程序。

```
root@OpenWrt:~# reboot
```

检测声卡驱动程序是否加载完成。

```
root@OpenWrt:~# ls /dev/dsp /dev/audio /dev/mixer
/dev/audio  /dev/dsp    /dev/mixer
```

以上信息都正确，那么恭喜，声卡驱动完成。

14.1.4 设置默认音量

默认音量比较大，可能首次播放会吓到小盆友们，需要修改下脚本设置音量。

命令行设置音量。

```
root@OpenWrt:/etc# amixer set PCM 35%
Simple mixer control 'PCM',0
  Capabilities: pvolume pswitch pswitch-joined
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 31
  Mono:
  Front Left: Playback 11 [35%] [11.00dB] [on]
  Front Right: Playback 11 [35%] [11.00dB] [on]
```

使用 Vi 编辑器编辑 /etc/rc.local 文件。在 exit 0 的前面增加设置语法。/etc/rc.local 修改后如下所示。

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

amixer set PCM 35%
exit 0
```

14.1.5 用命令行播放 MP3

安装命令行播放器 madplay。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install madplay
```

使用之前学过的文件复制方式，弄个 MP3 文件放到 /pub 下，文件名修改为 test.mp3。

使用命令启动播放 test.mp3。

```
root@OpenWrt:~# madplay /pub/test.mp3
```

欲停止播放，可通过组合键 Ctrl+C 中断播放工作。

14.2 安卓手机远程播放音乐

上一节中我们已经实现了对声卡的支持，现在我们使用安卓手机来控制 MR808 开发板播放音乐。

14.2.1 MP 介绍

MP 全称为音乐播放进程（Music Player Daemon）。它是一个可定制化、功能强大的服务端播放音乐软件，通过其自身的插件和库系统即可完成通过网络控制播放音乐。MP 具备一个控制管理协议，这些协议是基于 TCP 协议进行传输的，客户端连接到服务器上控制服务端进行处理。MPD 表示服务端，MPC 表示客户端。

MP 是一个完全开放源代码的整体系统，因此有很多操作系统下的客户端，在安卓手机

中可以使用 MPDroid 软件控制服务端播放音乐，如果你想要开发自己的产品，完全可以参考官方资料，编写一个控制程序，还可以给服务端增加新的功能。

使用 MP 的好处是只要你的手机连接到家里的路由器上，就可以控制播放音乐，而手机还可以随身携带，不会影响你做其他事情。

Tips

MP 官网：http://www.musicpd.org/

14.2.2 安装 MPD 服务

安装 MPD-MINI

安装软件包，本书中使用 MPD-MINI 软件包，它是一个 OpenWrt 系统下的精简版 MPD 服务，该软件包更节省 CPU 和内存资源。安装命令如下所示。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install mpd-mini
```

准备文件夹和文件

创建必须的文件夹，这一步，我们选择使用 /pub 作为存储位置，这样做的好处是上传到系统的文件可以通过 Windows 文件共享等方式来实现。当然你也可以开发一个新的客户端软件，实现将手机的音乐同步到服务端上。

创建音乐文件存放位置，并且设置权限。

```
root@OpenWrt:~# mkdir -p /pub/music
root@OpenWrt:~# chmod -R 777 /pub/music/
```

创建播放列表存放位置，并且设置权限。

```
root@OpenWrt:~# mkdir -p /pub/music_playlist
root@OpenWrt:~# chmod -R 777 /pub/music_playlist/
```

创建播放器数据位置。

```
root@OpenWrt:~# mkdir -p /pub/music_bin
root@OpenWrt:~# chmod -R 755 /pub/music_bin/
```

创建初始化的文件，这些文件都是 MPD 需要使用的。

```
root@OpenWrt:~# touch /pub/music_bin/mpd.db
root@OpenWrt:~# touch /pub/music_bin/mpd.log
```

```
root@OpenWrt:~# touch /pub/music_bin/mpd.error
root@OpenWrt:~# touch /pub/music_bin/mpdstate
root@OpenWrt:~# touch /pub/music_bin/mpd.sql
```

MPD 配置文件

编写配置文件，MPD 目前没有基于 UCI 格式的配置文件，需要手工创建一个 /etc/mpd.conf 文件。mpd.conf 配置文件参数说明如表 14-1 所示。

表 14-1 mpd.conf 配置文件参数说明

参数	说明	可选值及说明
music_directory	音乐文件的存放文件夹位置	目录路径
playlist_directory	音乐播放列表的存放位置，播放列表可以在设备上创建，也可以通过客户端软件创建	目录路径
db_file	MPD 的数据库文件路径	文件名路径
log_file	MPD 的日志文件路径	文件名路径
pid_file	MPD 的进程号文件路径	文件名路径
state_file	MPD 的状态文件路径	文件名路径
sticker_file	MPD 管理音乐动态信息的文件路径	文件名路径
user	MPD 运行时的 Linux 身份用户，默认 OpenWrt 下只有 root	root
group	MPD 运行时的 Linux 身份用户组，默认 OpenWrt 下只有 root	root
bind_to_address	MPD 监听的 IP 地址，0.0.0.0 表示全部设备的 IP	IP 地址，字符串
port	MPD 的监听端口	端口，字符串
log_level	日志信息输出的级别，可以填写 default、secure、verbose，其中 verbose 表示显示全部日志，信息很多	字符串
auto_update	当 music_directory 中的音乐文件有变化时自动更新 MPD 的数据库，可以填写 yes 或 no	字符串
auto_update_depth	自动更新的时候对文件夹深度的检测	数字
audio_buffer_size	音频播放时的缓存大小，默认值为 2048	数字
buffer_before_play	音频播放之前的缓存比例，默认值为 10%	百分比，字符串
connection_timeout	客户端连接超时，单位为秒，默认值为 60	数字
max_connections	客户端最大连接数量，默认值为 10	数字
max_playlist_length	播放列表文件允许的最大长度，这个不要弄得太大搞死系统了哦	数字
max_command_list_size	最大 list 指令尺寸，默认值为 2048	数字
max_output_buffer_size	最大输出缓存，默认值为 8192	数字
filesystem_charset	文件系统的编码，如果你的文件是通过 Windows 网络共享上载的，这里填写 UTF-8 即可	字符串

续表

参数	说明	可选值及说明
id3v1_encoding	歌曲的 ID3 标签名称编码，默认为“ISO-8859-1”	字符串
audio_output	MPD 的音频输出方式，这个配置实现的对声卡的支持，格式为 audio_output { 参数内容 }	字符串

mpd.conf 中 audio_output 支持的参数说明如表 14-2 所示。

```
audio_output {
        参数内容
}
```

表 14-2　audio_output 支持的参数说明

参数	说明	可选值及说明
type	输出类型，可选值为：oss、alsa、pulse、pipe	字符串
name	设备显示名称，在 MPDroid 客户端软件上可以显示出来	字符串
device	音频设备路径，默认填 /dev/sdp	设备路径，字符串
format	播放格式，默认填写为 44100:16:2 即可，表示 44kHz，16 位，双声道	字符串
mixer_type	混音器类型，我们的 MR808 开发板默认填写 hardware	字符串
mixer_device	混音器设备路径，我们的 MR808 开发板默认填写 /dev/mixer	设备路径，字符串
mixer_control	混音器的控制类型，我们的开发板默认填写 PCM	字符串

将旧的 mpd.conf 更名备份。

```
root@OpenWrt:~# mv /etc/mpd.conf /etc/mpd.conf.bak
```

使用 Vi 编辑器编写以下配置信息。

```
music_directory          "/pub/music"
playlist_directory       "/pub/music_playlist"
db_file                  "/pub/music_bin/mpd.db"
log_file                 "/pub/music_bin/mpd.log"
pid_file                 "/var/run/mpd.pid"
state_file               "/pub/music_bin/mpdstate"
sticker_file             "/pub/music_bin/mpd.sql"
user                      "root"
group                     "root"
bind_to_address           "0.0.0.0"
port                      "6600"
log_level                 "default"
auto_update          "yes"
auto_update_depth    "8"
```

```
audio_output {
        type    "oss"
        name    "MakerRouter_MR808"
        device  "/dev/dsp"
        format          "44100:16:2"
        mixer_type      "hardware"
        mixer_device    "/dev/mixer"
        mixer_control   "PCM"
}
#audio_buffer_size               "2048"
buffer_before_play              "10%"
connection_timeout               "60"
max_connections         "4"
max_playlist_length              "1024"
max_command_list_size            "2048"
max_output_buffer_size           "4096"
filesystem_charset              "UTF-8"
id3v1_encoding                   "ISO-8859-1"
```

复制音乐文件：上传音乐文件到 /pub/music 文件夹下，可以使用中文文件名，复制后更新列表需要一些时间，如果长时间没有发现上传的音乐文件，可以重启 MPD 和客户端。

修改 MPD 服务的默认音量：因为 MPD 服务的默认音量参数写得有问题，会导致音量被放大到 100%，声音太大，所以需要手工修改音量。修改方法如下所示。

用 Vi 编辑器编辑 /etc/init.d/mpd 文件，找到：

```
[ -x /usr/bin/amixer ] && /usr/bin/amixer set PCM 40
```

修改为：

```
[ -x /usr/bin/amixer ] && /usr/bin/amixer set PCM 40%
```

准备启动服务

如果经常使用，可以设置启动服务，每次开机自动启动。

```
root@OpenWrt:~# /etc/init.d/mpd enable
```

每次手动开启。

```
root@OpenWrt:~# /etc/init.d/mpd start
```

14.2.3 安卓系统安装客户端

MPDroid 是一个安卓系统下的免费客户端，该客户端可以实现大部分播放功能。

安装 MPDroid

1. 在网上搜索 MPDroid 并且下载安装。
2. 让手机连接到 MR808 开发板的 Wi-Fi 上。
3. 打开 MPDroid 主界面进行配置。配置界面如图 14-1、图 14-2 所示。

（1）选择 WLAN based connection。

（2）选择你的 MR808 开发板的 SSID。

（3）在 Host 中填写 MR808 开发板的 IP 地址，默认是 192.168.1.1。

（4）在 Port 中填写 MPD 监听的端口，默认是 6600。

（5）退出返回主界面。

图 14-1　MPDroid 配置界面 a

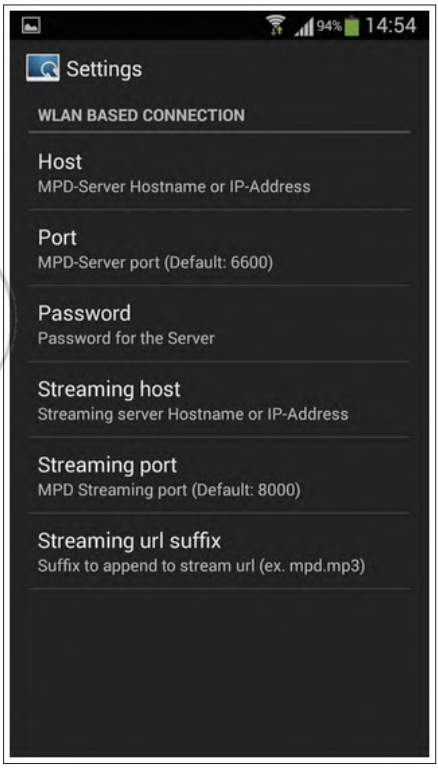


图 14-2　MPDroid 配置界面 b

4. 设置播放列表。

（1）单击主界面左上角的“Now Playing”，出现菜单选择“Library”。

（2）单击左上角的“Artists”，选择“Files”列表，显示出全部音乐文件，此时如果没有列表出现，请重新打开客户端软件或服务端软件。

（3）轻轻单击所有的文件，都单击过一遍之后按“菜单键”，选择“Save”，填写名

称，就可以生成播放列表，成功界面如图 14-3 所示。

5. 播放音乐

（1）再次单击左上角，选择“Now Playing”，会发现默认地已经进入了播放状态。

（2）单击“播放”按键，即可播放音乐，也可以更换音乐或控制音量等。播放音乐界面如图 14-4 所示。

■ 图 14-3 播放列表

■ 图 14-4 播放音乐

14.2.4 关于安卓手机播放音乐

MP 方式是一种很好的安卓播放技术，可惜有诸多不完善的地方，国内有一些厂商正在开发支持音乐播放的路由器，但为什么没有人考虑将这部分和在线音乐服务结合在一起呢？这样，当你回到家后就可以控制音响播放全部音乐，这是一件多么振奋人心的事情啊！

14.3 DLNA 数字媒体服务器

本节将介绍 DLNA 数字媒体服务器，虽然 OpenWrt 目前还不能实现 DLNA 直接播放音乐，但是作为数字音乐服务器，它可以为你全家的电子设备提供数字信息存储服务，也就相当于一个家庭私有云，可以存储电影、照片、音乐。而支持 DLNA 的设备十分广泛，有智能电视、安卓手机、Xbox、PS3、网络机顶盒等。

14.3.1 DLNA 介绍

DLNA 的全称是数字生活网络联盟（Digital Living Network Alliance），是一个由索尼、英特尔、微软等公司发起，旨在解决个人 PC、消费电器、移动设备的无线网络和有线网络的互联互通，使得数字媒体和内容服务的无限制的共享和增长成为可能，目前成员公司已达 280 多家。

DLNA 的标准已经深入各类智能家用电子设备中，其中安卓系统手机更是提供了必备的支持。本书使用的是一款三星安卓手机，手机中已经附带了 DLNA 的客户端程序。DLNA 并不是创造性的技术，而是一种解决方案，一种大家可以遵守的规范。所以，其选择的各种技术和协议都是当前应用非常广泛的。DLNA 分类如下所示。

- DMS：Digital Media Server（服务器端）用来存储数据，相当于媒体数据网络存储器。
- DMR：Digital Media Renderer（渲染器）用来接收数据。
- DMC：Digital Media Controller，控制端。
- DMP：Digital Media Player（播放器），用来播放接收到的数据。

OpenWrt 系统具备 DMS 功能，可以给予和发送数据到客户端，或是以流方式允许客户端播放流数据。安卓系统下的一些播放器已经具备了接收和服务功能，并且带有播放功能。

14.3.2 安装配置 MiniDLNA

MiniDLNA 是一个 DLNA 的 DMS 服务器。它可实现服务器组件，并给设备提供媒体文件信息。在 MiniDLNA 官方网站上是这样介绍的：“MiniDLNA 是一个简单的 DLNA 服务器，它是 NETGEAR 员工为他们的 ReadyNAS 产品线所开发的软件。”MiniDLNA 支持的文件类型有以下几种。

- **视频文件：**mpg、mpeg、avi、divx、asf、wmv、mp4、m4v、mts/m2ts/m2t、mkv、vob、ts、flv、xvid、TiVo（打开 tivo 支持）、mov、3gp。
- **音频文件：**mp3、flac、wma、asf、fla、flc、m4a、aac、mp4、m4p、wav、ogg、pcm、3gp。
- **图片文件：**jpg、jpeg。
- **列表文件：**m3u、pls。

安装 MiniDLNA。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install minidlna
```

配置 MiniDLNA，/etc/config/minidlna 配置文件内容如下所示。

```
config minidlna config
        option 'enabled' '1'
```

```
        option port '8200'
        option interface 'br-lan'
        option friendly_name 'OpenWrt DLNA Server'
        option db_dir '/var/run/minidlna'
        option log_dir '/var/log'
        option inotify '1'
        option enable_tivo '0'
        option strict_dlna '0'
        option presentation_url ''
        option notify_interval '900'
        option serial '12345678'
        option model_number '1'
        option root_container '.'
        list media_dir '/mnt'
        option album_art_names 'Cover.jpg/cover.jpg/AlbumArtSmall.jpg/
albumartsmall.jpg/AlbumArt.jpg/albumart.jpg/Album.jpg/album.jpg/Folder.
jpg/folder.jpg/Thumb.jpg/thumb.jpg'
```

UCI 查看方法如下。

```
root@OpenWrt:/etc/config# uci show minidlna
minidlna.config=minidlna
minidlna.config.enabled=1
minidlna.config.port=8200
minidlna.config.interface=br-lan
minidlna.config.friendly_name=OpenWrt DLNA Server
minidlna.config.db_dir=/var/run/minidlna
minidlna.config.log_dir=/var/log
minidlna.config.inotify=1
minidlna.config.enable_tivo=0
minidlna.config.strict_dlna=0
minidlna.config.notify_interval=900
minidlna.config.serial=12345678
minidlna.config.model_number=1
minidlna.config.root_container=.
minidlna.config.media_dir=/mnt
minidlna.config.album_art_names=Cover.jpg/cover.jpg/AlbumArtSmall.jpg/
albumartsmall.jpg/AlbumArt.jpg/albumart.jpg/Album.jpg/album.jpg/Folder.
jpg/folder.jpg/Thumb.jpg/thumb.jpg
```

minidlna.config 选项的参数说明如表 14-3 所示。

表 14-3 minidlna.config 选项参数说明

选项	说明	可选值及说明	必填
enable	是否允许启动 minidlna	0：否 1：是	是
port	dlna 服务端口，默认为 8200	数值	是

续表

选项	说明	可选值及说明	必填
interface	工作在哪个网络接口上，br-lan 表示工作在 LAN 口和 Wi-Fi 上	字符串	是
friendly_name	在客户端设备上看到的名字	字符串	是
db_dir	minidlna 数据库文件路径	路径，字符串	是
log_dir	日志文件路径	路径，字符串	是
inotify	设置是否自动检测新文件，这个参数跟 overlay 挂载有点冲突，下面我们再详细说明	路径，字符串	是
enable_tivo	是否支持 Tivo 标准，这个是一个老式录像设备上经常支持的标准	0：否 1：是	是
strict_dlna	是否允许传输超大的 JPEG 图像，在客户端是索尼的产品时会消耗很大的性能	0：否 1：是	是
presentation_url	是否设置一个 URL 地址和端口，一般不用配置		是
notify_interval	自动检测间隔时间，如果不频繁更新，可以设置得慢一些	数值，单位为秒	是
serial	序列号，用来提供给客户端的，一般保持默认即可	数值	是
model_number	型号，用来提供给客户端的，一般保持默认即可	数值	是
root_container	客户端能看到的选项，可以填写多个，以逗号“,”分隔	.：默认 B：文件夹 M：音频 V：视频 P：图片	是
media_dir	media_dir 是一个列表键，存放媒体文件的路径地址	路径，字符串	是
album_art_names	专辑图片信息的搜索文件名。一般就是专辑封面什么的	字符串	是

14.3.3 Inotify 与 overlay 的冲突

本系统采用透明挂载实现了外置存储器，这种方式与 minidlna 存在不兼容，具体表现为：当媒体目录有新文件更新时无法检测到更新，解决办法是直接填写实际路径（例如我们使用 /pub/dlna，那么我们在配置时填写 /overlay/pub/dlna 即可完成）。

配置举例如下。

```
root@OpenWrt:~# uci del_list minidlna.config.media_dir='/mnt'
root@OpenWrt:~# uci set minidlna.config.enabled='1'
root@OpenWrt:~# uci set minidlna.config.port='8200'
root@OpenWrt:~# uci set minidlna.config.interface='br-lan'
root@OpenWrt:~# uci set minidlna.config.friendly_name='MakerRouter DLNA'
root@OpenWrt:~# uci set minidlna.config.db_dir='/var/run/minidlna'
root@OpenWrt:~# uci set minidlna.config.log_dir='/var/log'
root@OpenWrt:~# uci set minidlna.config.inotify='1'
root@OpenWrt:~# uci set minidlna.config.enable_tivo='0'
root@OpenWrt:~# uci set minidlna.config.strict_dlna='0'
root@OpenWrt:~# uci set minidlna.config.presentation_url=''
```

```
root@OpenWrt:~# uci set minidlna.config.notify_interval='10'
root@OpenWrt:~# uci set minidlna.config.serial='12345678'
root@OpenWrt:~# uci set minidlna.config.model_number='1'
root@OpenWrt:~# uci set minidlna.config.root_container='.'
root@OpenWrt:~# uci add_list minidlna.config.media_dir='/overlay/pub/dlna'
root@OpenWrt:~# uci set minidlna.config.album_art_names='Cover.jpg/
cover.jpg/AlbumArtSmall.jpg/albumartsmall.jpg/AlbumArt.jpg/albumart.
jpg/Album.jpg/album.jpg/Folder.jpg/folder.jpg/Thumb.jpg/thumb.jpg'
root@OpenWrt:~# uci commit minidlna
```

创建 /pub/dlna 目录。

```
root@OpenWrt:~# mkdir -p /pub/dlna
root@OpenWrt:~# chmod -R 777 /pub/dlna
```

14.3.4 启动服务

复制文件

现在我们可以向 /pub/dlna 中复制文件或文件夹，文件支持的类型在前面已经讲过，凡是支持的类型都将能被客户端查找和发现。

启动服务

设置开机自动启动服务。

```
root@OpenWrt:~# /etc/init.d/minidlna enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/minidlna start
```

关闭 minidlna，由于 MiniDLNA 驱动脚本有故障，这里关闭 MiniDLNA 使用进程杀死方式。

```
root@OpenWrt:~# killall minidlna
```

14.3.5 DLNA 客户端

针对不同的设备，例如智能电视机、机顶盒等，有不同类型的 DLNA 客户端，本系统使用一个第三方的音乐播放器客户端 BubbleUPnP，来实现播放服务器中的音乐文件。实现过程如下。

在网上搜索 BubbleUPnP，并且下载安装，让手机连接到 MR808 开发板的 Wi-Fi 上。打开 BubbleUPnP 的主界面，配置如下所示。

（1）单击屏幕左上角的图标显示出菜单。

（2）选择 LIBRARY 下的“Local Media Server”，在新弹出菜单中选择“MakerRouter DLNA”，配置界面如图 14-5 所示。

（3）在 Media 下选择“Browser Folders”，配置界面如图 14-6 所示。

（4）屏幕中既可以选择播放的音乐文件，也可以下载该文件到手机上，音乐播放界面如图 14-7 所示。

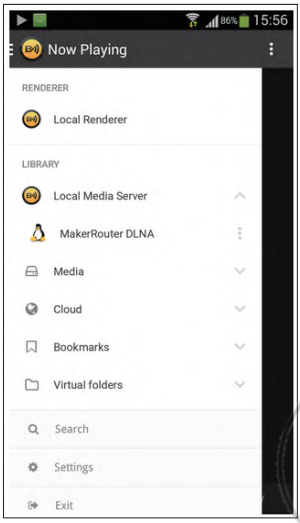


图 14-5 选择“MakerRouter DLNA”

图 14-6 “Browser Folders”界面

图 14-7 音乐播放界面

14.3.6 DLNA 其他功能

众所周知，现在机顶盒和电视机都已经具备了 DLNA 接口，只要将电影下载到 MR808 开发板的 /pub/dlna 目录下，即可被媒体设备自动发现，从而实现播放。如果配合我们后续章节介绍的迅雷离线下载，可以简单组成家庭播放数字中心。

14.4 苹果 AirPlay 播放音乐

14.4.1 AirPlay 介绍

AirPlay 是美国苹果公司的在 iOS 4.2 及 OS X Mountain Lion 中加入的一种播放技术，可以将 iPhone、iPod touch、iPad 及 Mac（需要 OS X Mountain Lion）上的视频镜像传送到支持 Airplay 的设备（如音箱、Apple TV）中播放。使用 Airplay 需要 iOS 4.2 以上版本的设备或 Mac 计算机上的 iTunes 10.1 以上版本。视频镜像则需要 iOS 5 及 OS X Mountain Lion。

苹果 AirPlay 在播放时可以自动发现局域网中的播放设备，然后选择用其进行播放。这样流数据就可以通过 Wi-Fi 直接发送到音箱（运行 OpenWrt 的智能设备）上进行播放。

14.4.2 安装与配置 Shairport

我们使用了一款叫作 ShairPort 的开源软件作为 AirPlay 的服务端。ShairPort 安装配置过程如下。

1. 安装 Shairport

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install shairport
```

安装完成就 OK 了，要确保音箱声卡都连接完成，并且之前章节的测试也要 OK。

2. Shairport 配置文件

/etc/config/shairport 的配置内容如下。

```
config shairport main
        option bname 'AirPort'
        option password ''
        option port '5002'
        option buffer '256'
        option log '/var/log/shairport'
        option cmd_start ''
        option cmd_stop ''
        option cmd_wait ''
        option audio_output 'alsa'
```

```
        option mdns 'avahi'

        # options for alsa output
        option output_dev ''
        option mixer_dev ''
        option mixer_type ''
        option mixer_control ''
        option mixer_index ''
```

UCI 查看方法如下。

```
root@OpenWrt:/etc/config# uci show shairport
shairport.main=shairport
shairport.main.bname=AirPort
shairport.main.port=5002
shairport.main.buffer=256
shairport.main.log=/var/log/shairport
shairport.main.audio_output=alsa
shairport.main.mdns=avahi
```

shairport.main 节点选项的参数说明如表 14-4 所示。

表 14-4　shairport.main 节点选项参数说明

选项	说明	可选值及说明	必填
bname	在苹果系统中看到的名字	字符串	是
port	服务端口，默认为 5002	端口，数值	是
buffer	播放时使用的缓冲区大小	数值，范围为 32~512，单位为帧	是
log	日志文件位置	文件路径，字符串	否
audio_output	音频输出方式类型，默认为 alsa	alsa 或 oss	是
mdns	组播 DNS，在 OpenWrt 下使用的是 avahi，默认为 avahi	字符串	是
output_dev	设置选择哪个输出设备，只有一个声卡时不用配置	default 或设备，字符串	否
mixer_dev	混音器设备，按默认即可，除非混音器设备特殊	设备名称，字符串	否
mixer_type	混音方式	hardware：硬件 software：软件	否
mixer_control	混音控制类型，默认为 Master（主音频）	混音方式名称，字符串	否
mixer_index	混音器序列号	0 为默认序列号，数值	否

3. Shairport 配置

默认情况下，不用对混音器做任何配置，通过默认配置即可找到声卡设备，也可以修改 bname 设置，设置苹果设备上看到的名字和开启 log 文件。

4. 启动 Shairport

手动启动 dbus 关联服务，第一次启动会报一个警告，启动两次后正常。

```
root@OpenWrt:~# /etc/init.d/dbus start
root@OpenWrt:~# /etc/init.d/dbus start
```

手动启动 avahi 关联服务。

```
root@OpenWrt:~# /etc/init.d/avahi-daemon start
```

手动启动 Shairport。

```
root@OpenWrt:~# /etc/init.d/shairport start
```

设置自动启动服务。

```
root@OpenWrt:~# /etc/init.d/dbus enable
root@OpenWrt:~# /etc/init.d/avahi-daemon enable
root@OpenWrt:~# /etc/init.d/shairport enable
```

14.4.3 辅助配置和检查

查看 Shairport 是否启动。

```
root@OpenWrt:~# netstat -lnp|grep shairport
tcp        0      0 0.0.0.0:5002            0.0.0.0:*
LISTEN      1565/shairport
tcp        0      0 :::5002                 :::*
LISTEN      1565/shairport
```

查看 Shairport 关于配置参数的详细说明。

```
root@OpenWrt:~# shairport --help
```

爆音和音颤：有的声卡可能存在爆音和音颤的问题，使用下面方法可以改善。

使用 Vi 编辑器编辑文件 etc/asound.conf，内容如下。

```
defaults.pcm.dmix.rate 44100
defaults.pcm.dmix.format S16_LE
```

alsa 的默认配置为：强制 44.1kHz，强制 16 位。

重启系统使之生效。

```
root@OpenWrt:~# reboot
```

14.4.4 使用终端播放

将具备播放能力的设备（iPhone 手机、iPad 平板电脑、Mac 计算机、Windows 计算机）通过 LAN 或无线方式连接到 MR808 开发板上。

- iPhone 手机：使用手机的播放器播放本地保存的音乐，屏幕向上滑动出现音量调节界面，选择 AirPlay 后，选择播放设备会有一个设备名字就是 bname 的那个名字，选中设备后，就可以通过音箱播放音乐了。
- iPAD 平板电脑：操作方法类似 iPhone 手机。
- Mac 计算机：使用 iTunes 播放器，在“播放”的边上有一个小图标可以实现 iPhone 一样的功能。
- Windows 计算机：下载使用 iTunes 播放器的 Windows 版本，操作方法与 Mac 版相同。

第15章 智能路由器编程工具

15.1 Web+PHP 建站系统

在 x86 的服务器上有很多应用系统或是网站都是基于 LAMP（Linux + Apache + MySQL + PHP）开发的。在智能路由器中也完全可以搭建一套类似的应用系统或网站，本系统采用 OLSP（OpenWrt+Lighttpd+Sqlite+PHP）来完成搭建。

本章还介绍了 MySQL 数据库、Sqlite3 数据库、Python 语言、Lua 语言的安装和基本使用方式。使用这些功能丰富的工具可以非常好地帮助开发人员进行智能路由器开发，让开发人员几乎忘掉所开发的程序是运行在路由器上的。这也体现出了智能化设备与传统功能型路由器的区别。

15.1.1 PHP 语言

1. PHP 介绍

PHP 中文名字为“超文本预处理器”（Hypertext Preprocessor），是目前全球互联网上最火爆的 Web 后端语言，拥有大量的开源库，其特点主要有： 运行快、学习简单、库资源丰富。一般具备任何一门编程语言背景的朋友只要花费上一两个小时便能入门，一周时间基本可以掌握开发技巧。在 OpenWrt 系统下，可以采用 FastCGI 技术让 PHP 和 Lighttpd 配合工作，其性能和内存资源占用率都非常均衡，而且网上有很多系统，比如著名的论坛系统 Discuz，早已有人将其安装到 OpenWrt 中。

当然有人会说：“Java 不是最流行的么？”确实，Java 在中国很流行，但是 Java 并不是最适合的嵌入式编程语言，其主要原因是 Java 语法亢奋、虚拟机运行效率低下、跨平台苦难重重。

2. 安装 PHP

查看本系统支持多少个 PHP 模块。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg find php5-*
```

推荐 PHP 爱好者选择安装自己心仪的 PHP 包，这里仅是举例安装，不必一致。

```
root@OpenWrt:~# opkg install php5-cli php5-cgi php5-fastcgi php5-mod-
session php5-mod-sqlite3 php5-mod-mysql php5-mod-ctype php5-mod-gd
php5-mod-sockets php5-mod-mcrypt php5-mod-mbstring php5-mod-curl
```

用命令行测试一下 PHP。

```
root@OpenWrt:~# php-cli --version
PHP 5.4.27 (cli) (built: Feb 28 2015 18:26:36)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2014 Zend Technologies
```

PHP 在 OpenWrt 下的主要解析器如下。

- php-cli：在命令行模式下执行 PHP 程序的解析器。
- php-cgi：在 CGI 模式下执行 PHP 程序的解析器。
- php-fcgi：在 FastCGI 模式下执行 PHP 程序的解析器。

15.1.2 Web 服务

1. Lighttpd 介绍

在服务器上的 Linux 系统中，我们普遍使用的 Web 服务器有两种：一种是 Nginx，另一种是 Apache，它们都是非常著名的系统，在全球使用量在上百万套，它们也拥有 OpenWrt 下的版本，但是由于体系庞大、支持复杂，并不适合我们这种小设备使用。就好比你也不会买辆微型面包车，配一台 3.6L 的发动机吧?

Lighttpd 是一个由德国人发起的开源 Web 服务器软件，其特点是高性能、安全、快速、兼容性好、内存开销低、CPU 占用率低等。Lighttpd 是所有轻量级 Web 技术中最优秀的服务器软件之一。它支持 FastCGI、 CGI、Auth、Compress、Rewrite、Alias 等必备的 Web 服务器技术。

PHP 采用 FastCGI 模式与 Lighttpd 连接，内存中驻留一定数量的进程等待请求，在消耗一定内存的情况下，降低了 CPU 占用率，提升了性能。除此之外，OpenWrt 的 LUCI 项目采用的是 uhttp+uhttp_Lua（嵌入式 Lua）+libuci 语言开发完成的，在后续的章节中我们也将介绍 Lua 语言的安装，不过该语言并不适用于功能复杂的建站环境。

2. 安装 Lighttpd

查看教学系统支持多少个 Lighttpd 模块。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg find lighttpd-*
```

安装 Lighttpd 常见软件包。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install lighttpd lighttpd-mod-fastcgi lighttpd-
mod-access lighttpd-mod-alias lighttpd-mod-redirect lighttpd-mod-
rewrite lighttpd-mod-evasive
```

准备 Web 文件存放路径。

```
root@OpenWrt:~# mv -f /www /pub/www
root@OpenWrt:~# chmod -R 777 /pub/www
root@OpenWrt:~# ln -s /pub/www/ /www
root@OpenWrt:~# ls -l ../www
lrwxrwxrwx    1 root     root             9 Mar  1 02:45 ../www -> /pub/www/
```

3. 配置 Lighttpd

Lighttpd 的配置文件不支持 UCI 模式，Lighttpd 的配置文件为 /etc/lighttpd/lighttpd.conf，其参数说明如表 15-1 所示。

表 15-1 lighttpd.conf 配置文件参数说明

参数	说明	可选值及说明
server.modules	启用哪些 Lighttpd 模块	已安装模块名称
server.document-root	设置 www 主文件夹地址	目录路径
server.errorlog	设置错误信息文件路径，如果不设置这个参数就表示禁用	文件名路径
server.port	设置 Web 端口，默认为 80	端口，数值
index-file.names	设置索引主文件名称	字符串
static-file.execute-extesions	设置可执行文件的名称	字符串
server.pid-file	设置 Lighttpd 服务的 pid 文件存放名称	文件名路径
server.upload-dirs	设置临时 Web 上传文件位置	目录路径
evasive.max-conns-per-ip	设置同时允许多少个请求一起发给 Lighttpd，这个参数是 mod_evasive 模块提供的	字符串

备份旧的 lighttpd.conf 文件

```
root@OpenWrt:~# mv /etc/lighttpd/lighttpd.conf /etc/lighttpd/lighttpd.
conf.bak
```

关于 FastCGI 模式，在 lighttpd.conf 中即可直接配置 PHP 的 FastCGI 模式工作，支持参数如表 15-2 所示。

表 15-2　配置 FastCGI 模式说明

参数	说明	可选值及说明
bin-path	FastCGI 解析器，就是 /usr/bin/php-fcgi	字符串
bin-environment	Lighttpd 给予输出的环境变量，在 PHP 下可以接收 PHP_FCGI_CHILDREN 产生进程数量 PHP_FCGI_MAX_REQUESTS 最大请求	字符串
min-procs	最小进程数量	数值
max-procs	最大进程数量	数值
idle-timeout	空闲超时时间	数值，单位为秒

4. 配置 Lighttpd

使用 Vi 编辑器编写新的 /etc/lighttpd/lighttpd.conf，内容如下。

```
server.modules = (
        "mod_access",
        "mod_alias",
        "mod_redirect",
        "mod_rewrite",
        "mod_fastcgi",
        "mod_evasive"
)

server.document-root = "/pub/www"
server.errorlog = "/var/log/lighttpd/error.log"
server.port = 80
index-file.names = ( "index.php", "index.html", "index.htm" )
static-file.exclude-extensions = ( ".php" )
server.pid-file = "/var/run/lighttpd.pid"
server.upload-dirs = ( "/tmp" )
evasive.max-conns-per-ip = 18

fastcgi.server = ( ".php" =>
  ((
      "socket" => "/tmp/php-fastcgi.socket",
      "bin-path" => "/usr/bin/php-fcgi",
      "bin-environment" => (
          "PHP_FCGI_CHILDREN" => "2",
          "PHP_FCGI_MAX_REQUESTS" => "100"
      ),
      "min-procs" => 1,
      "max-procs" => 2,
      "idle-timeout" => 20
   ))
)
```

5. 启动 Lighttpd

设置启动服务。

```
root@OpenWrt:~# /etc/init.d/lighttpd enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/lighttpd start
```

15.1.3 PHP 测试代码

因为路径在 /pub/www 下，所以我们可以很方便地使用 Windows 网上邻居这种方式来编写代码，也可以在 OpenWrt 环境中使用 Vi 编写代码。程序的扩展名为 .php 即可被执行。

1. PHP 信息输出测试

编辑文件 /pub/www/test1.php，内容如下。

```
<?php
phpinfo();
?>
```

PHP 信息输出测试界面如图 15-1 所示。

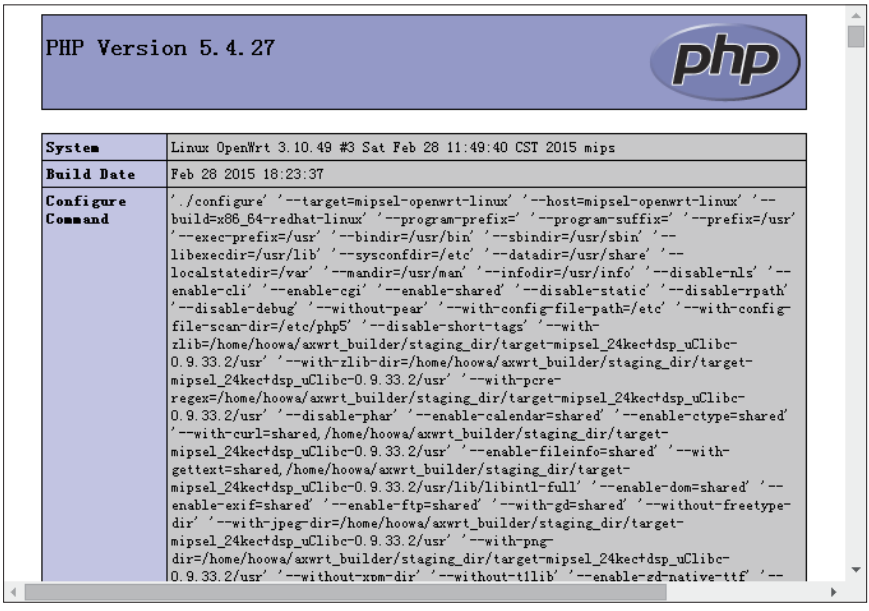


图 15-1 PHP 信息输出测试界面

2. PHP 显示网络配置

使用 PHP 语言可以执行外部 UCI 命令获得当前配置参数，本例中我们读取 3 个地方信息，如下所示。

- network.lan
- network.wan
- network.wireless.@wifi-iface[0]

编辑文件 /pub/www/get_network.php，内容如下。

```
<?php
$lan = `uci show network.lan`;
$wan = `uci show network.wan`;
$wifi = `uci show wireless.@wifi-iface[0]`;
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <title> get_network </title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 </head>

 <body>
<H3> 外网配置信息 </H3>
<pre>
<?=$lan?>
</pre>

<H3> 局域网配置信息 </H3>
<pre>
<?=$wan?>
</pre>

<H3> 无线口配置信息 </H3>
<pre>
<?=$wifi?>
</pre>

 </body>
</html>
```


在浏览器中访问 MR808 开发板的 IP 地址，例如：http://192.168.1.1/get_network.php，测试界面如图 15-2 所示。

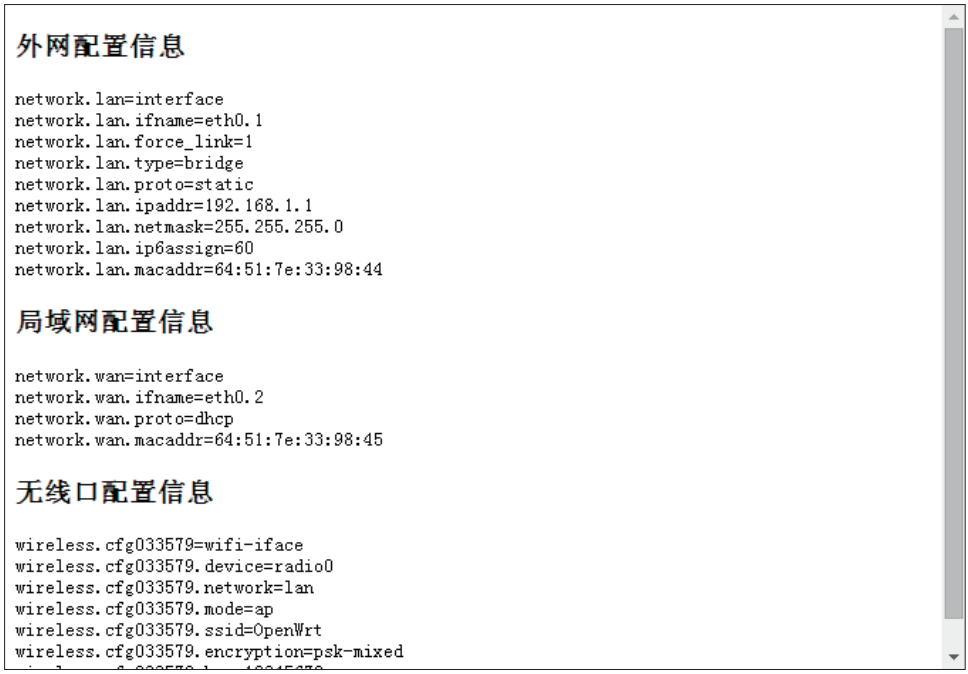


图 15-2 在浏览器中访问 MR808 开发板的 IP 地址

15.2 SQLite3 与 MySQL 数据库应用

15.2.1 SQLite3 数据库

1. SQLite3 介绍

SQLite 是一款轻量级的关系型数据库，它的设计目标就是嵌入式，它占用系统资源非常少，甚至连进程都没有，且数据结构的敏感程度也非常低，但是它的语法与其他流行的关系数据库，比如 MySQL 是非常接近的，因此很适合用在 OpenWrt 系统中。在本系统中保留了各种语言对 SQlite3 的支持。

由繁入简，不论采用何种数据库技术，最终归根结底都是要对 I/O 进行操作，只有当一种技术能最直接、最有效地操作 I/O，才是最需要的技术，才最适合嵌入式应用，SQLite 便是！SQLite 3 是目前最新版的 SQLite 技术，增加了很多处理和保护机制。

Tips

SQLite 官网学习资料：http://www.sqlite.org

2. SQLite 误区：

（1）**SQLite 也有服务**：SQLite 没有服务，SQLite 总是嵌入各种编程语言当中，不需

要连接到一个所谓的数据库服务。

（2）**SQLite 的数据库创建**：确实可以在 SQLite 中创建数据库，但事实上是那只是个文件，SQLite 都是直接针对文件操作的。

（3）**SQLite 速度非常慢**：这是一个典型的误区，SQLite 采用 I/O 口直接操作，是迄今性能最高的数据库之一，SQLite 的缺点在于同时访问，因为针对 I/O 口操作存在文件锁的问题。

（4）**SQLite 文件锁**：SQLite 可以解决这个问题，就是把多个表分离成不同的库（文件），文件多了就好办了。

（5）**SQLite 的数据稳定性**：SQLite 在数据稳定性上要比传统数据库更强，尤其是拥有断电异常及保护机制，绝对不会像传统数据库那般脆弱。

3. SQLite3 命令

上文已提到，实际上不需要单独安装所谓的 SQLite 服务，各种语言中已经包含了相关的库，直接调用就好了，它是嵌入式的。在这里我们介绍安装 SQLite 命令，这个软件的作用是在命令行下查看或管理数据库文件。

通过命令行安装。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install sqlite3-cli
```


命令语法格式如下。

```
sqlite3 [ 库文件名 ]
```

举例：创建数据库（进入 sqlite 命令模式）。

```
root@OpenWrt:~# sqlite3 /pub/test_sqlite3.db
SQLite version 3.8.7.4 2014-12-09 01:34:36
Enter ".help" for usage hints.
sqlite>
```

准备一个含有字段 name 和字段 age 的表，表创建语法如下。

```
create table test(
name varchar(60),
age int(3)
);
```

举例：创建表，在 sqlite> 命令行下输入创建语法，结尾的分号（;）表示执行。

```
sqlite> create table test(
```

```
   ...> name varchar(60),
   ...> age int(3)
   ...> );
sqlite>
```

举例：插入数据与查看。

```
sqlite> insert into test(name,age) values('hoowa','18');
sqlite> insert into test(name,age) values('zhanglu','90');
sqlite> select * from test;
```

举例：退出。

```
sqlite> .exit
```

查看数据库文件。

```
root@OpenWrt:~# ls -l /pub/test_sqlite3.db
-rw-r--r--    1 root     root          2048 Mar  2 03:26 /pub/test_
sqlite3.db
```

通过上面命令就可以完成表的正常操作，使用感受同 MySQL 数据库相当。

4. PHP 使用 SQLite3

下面介绍 PHP 语言如何使用 SQLite3，本系统所支持的其他语言都支持 SQLite3 操作，相关介绍将在下一章进行。关于 PHP 语言如何使用 SQLite3 函数和接口的详细介绍请访问 www.php.net 获得官方说明。

安装 PHP 的 SQLite3 支持。

```
root@OpenWrt:~# opkg install php5-mod-sqlite3
```

举例：实现一个查询数据库的 Web 程序 /pub/www/sqlite3.php，执行后，查询显示界面如图 15-3 所示。

```
<?php

// 以读写形式打开数据库文件，如果文件不存在就尝试创建
$db = new SQLite3("/pub/test_sqlite3.db",SQLITE3_OPEN_READWRITE |
SQLITE3_OPEN_CREATE);

// 设置 sqlite 存储编码
$db->query('PRAGMA encoding="UTF-8"');
// 设置 sqlite 同步缓存方式
$db->query('PRAGMA synchronous = NORMAL');
```

```
// 查询
$result=@$db->query("select * from test");
if ($db->lastErrorCode() !== 0) {
      // 如果查询有错则显示错误
      echo "Error: ".$db->lastErrorMsg();
      exit;
}

// 准备输出
$output="";
while ($row = $result->fetchArray(SQLITE3_ASSOC)) {
      $output.="<p><b>name: </b> ".$row['name'];
      $output.=" <b>age: </b> ".$row['age']."</p>";
}
$result->finalize(); // 输出结束

// 关闭数据库
$db->close();

?>
<html>
 <head>
  <title> MakerRouter </title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 </head>

 <body style="background-color: #f3f3f3">
<?=$output;?>
 </body>
</html>
```

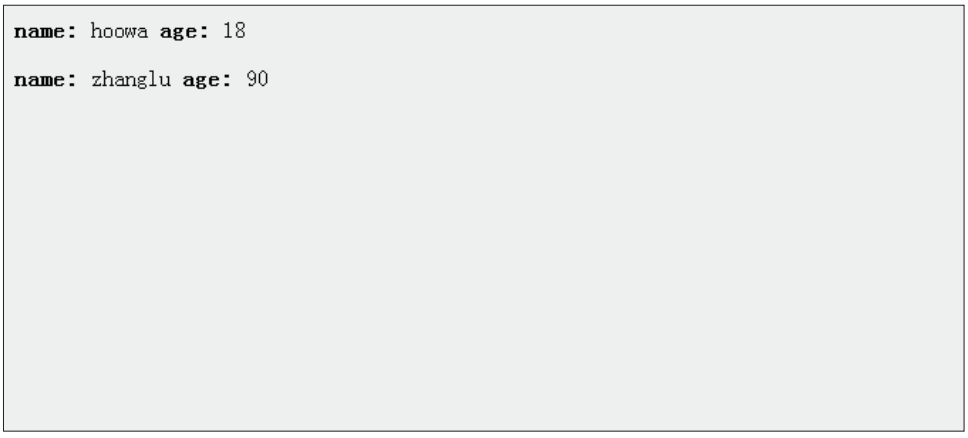



图 15-3 查询显示界面

15.2.2 MySQL 数据库

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司。MySQL 是最流行的关系型数据库管理系统，在 Web 应用方面，MySQL 是最好的 RDBMS（Relational Database Management System，关系型数据库管理系统）应用软件之一。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。由于其社区版的性能卓越，在服务器环境中搭配 PHP 可组成良好的开发环境。

Tips

MySQL 官网：http://www.mysql.com

在 MR808 开发板中运行 MySQL 并不是明智的选择，运行速度较慢。但如果想移植一些在服务器上运行的程序，比如著名的 Discuz 论坛，那么这是一个选择。

1. MySQL 的安装与配置

不同于嵌入式的 SQLite3，MySQL 需要独立安装、开启服务才能正常启动使用。

安装 MySQL。

```
root@OpenWrt:~# opkg install mysql-server
```

MySQL 的库存储位置一般在 /var/lib 下，由于 OpenWrt 的 var 文件夹是内存临时文件目录，所以我们要重新定义这部分路径信息。

准备库文件存储位置。

```
root@OpenWrt:~# mkdir -p /mnt/data /mnt/data/mysql /mnt/data/tmp
```

初始化默认的 MySQL 数据。

```
root@OpenWrt:~# mysql_install_db --force
```

2. MySQL 的启动

设置启动服务。

```
root@OpenWrt:~# /etc/init.d/mysql enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/mysql start
```

3. PHP 使用 MySQL

MySQL 数据库的使用方法在网络上有大量的学习资料，很多爱好者也都擅长使用该数

据库。这里做一个实例演示。

安装 PHP 的 MySQL 支持。

```
root@OpenWrt:~# opkg install php5-mod-mysql
```

举例：实现一个查询数据库的 Web 程序 /pub/www/mysql.php。

```
<?php

// 简单演示 MySQL 的连接是否成功
$link = mysql_connect('127.0.0.1', 'root', '');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);

?>
```

15.3 Python、Lua 与 Ruby 编程工具

在广大爱好者的强烈要求下，本章简单介绍几种编程语言的安装方法，在本系统固件的源码中，这几种语言已经默认被选中。

15.3.1 Python

Python 是一种面向对象的解释型计算机程序设计语言，由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。Python 语法简洁而清晰，具有丰富和强大的类库。它常被昵称为胶水语言，能够很轻松地把用其他语言（尤其是 C/C++）制作的各种模块联结在一起。

在 OpenWrt 环境下可以使用 Python 语言开发一些应用程序，以及通过 CGI 方式配置实现和 Web 服务协同工作，但某些扩展库存在移植困难，支持并不是非常完善。下面介绍如何安装 Python。

查找全部 Python 软件包。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg find python*
```

可以根据需要来安装软件包，这里举例安装基本包。

```
root@OpenWrt:~# opkg install python
```

测试 Python 是否已经存在。

```
root@OpenWrt:~# python --version
Python 2.7.3
```

15.3.2 Lua

Lua 是一个小巧的脚本语言，是巴西里约热内卢天主教大学（Pontifical Catholic University of Rio de Janeiro）里，由 Roberto Ierusalimschy、Waldemar Celes 和 Luiz Henrique de Figueiredo 所组成的一个研究小组于 1993 年开发的。其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。Lua 由标准 C 编写而成，在绝大多数操作系统和平台上都可以编译、运行。Lua 并没有提供强大的库，这是由它的定位决定的。所以 Lua 不适合作为开发独立应用程序的语言。Lua 有一个同时进行的 JIT 项目，提供在特定平台上的即时编译功能。

OpenWrt 环境下对 Lua 的支持非常好，大名鼎鼎的 LUCI 项目（基于 uhttpd+Lua+libuci 开发）便是基于 Lua 语言设计，但由于 Lua 语言过于简单和薄弱，使用其开发一个复杂的 Web 系统将会成为噩梦。下面介绍如何安装 Lua。

查找全部 Lua 的软件包。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg find lua*
```

可以根据需要安装软件包，这里举例安装基本包。

```
root@OpenWrt:~# opkg install lua
```

测试 Lua 是否已经存在。

```
root@OpenWrt:~# lua -v
Lua 5.1.5  Copyright (C) 1994-2012 Lua.org, PUC-Rio (double int32)
```

15.3.3 Ruby

Ruby 是一种为简单、快捷的面向对象编程（面向对象程序设计）而创的脚本语言，在 20 世纪 90 年代由日本人松本行弘（まつもとゆきひろ /Yukihiro Matsumoto）开发。它的灵感与特性来自于 Perl、Smalltalk、Eiffel、Ada 以及 Lisp 语言。Ruby 的作者于 1993 年 2 月 24 日开始编写 Ruby，直至 1995 年 12 月才正式公开发布于 fj（新闻组）。

在 OpenWrt 环境下可以使用 Ruby 语言开发一些应用程序，以及通过 CGI 方式配置实现跟 Web 服务协同工作，但某些扩展库存在移植困难，支持并不是非常完善。下面介绍如何安装 Ruby。

查找全部 Ruby 的软件包。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg find ruby*
```

可以根据需要安装软件包，这里举例安装基本包。

```
root@OpenWrt:~# opkg install ruby ruby-cgi
```

测试 Ruby 是否已经存在。

```
root@OpenWrt:~# ruby --version
ruby 1.9.3p545 (2014-02-24) [mipsel-linux]
```

第16章 智能路由器服务软件

16.1 VPN（PPTP）服务器与客户端

VPN（虚拟专用网）被定义为通过一个公用网络（通常是国际互联网）建立一个临时的、安全的连接，是一条穿过混乱的公用网络的安全、稳定的隧道。 虚拟专用网是对企业内部网的扩展。PPTP（Point to Point Tunneling Protocol），即点对点隧道协议，是在 Windows 系列操作系统以及各类智能终端设备中广泛支持的 VPN 协议标准。

16.1.1 PPTP 协议 VPN 服务器

在智能路由器中建立 VPN 服务器即可允许外网使用者连接进入内网。下面介绍如何安装 PPTP 软件包。

安装 PPTP 软件包。

```
root@OpenWrt:~# opkg install pptpd kmod-mppe
```

安装完成后要重启系统使驱动生效。

```
root@OpenWrt:~# reboot
```

修改默认的 /etc/init.d/pptpd，该文件有 BUG，会导致连接不支持多客户端，修改内容如下。

```
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org

START=60
BIN=/usr/sbin/pptpd
DEFAULT=/etc/default/$BIN
RUN_D=/var/run
PID_F=$RUN_D/$BIN.pid
CONFIG=/var/etc/pptpd.conf
CHAP_SECRETS=/var/etc/chap-secrets

setup_login() {
      local section="$1"
```

```
	config_get username "$section" username
	config_get password "$section" password
	config_get remoteip "$section" remoteip
	[ -n "$username" ] || return 0
	[ -n "$password" ] || return 0
	[ -n "$remoteip" ] || return 0

	echo "$username pptp-server $password $remoteip" >> $CHAP_SECRETS
}

setup_config() {
	local section="$1"

	config_get enabled "$section" enabled
	[ "$enabled" -eq 0 ] && return 1

	mkdir -p /var/etc
	cp /etc/pptpd.conf $CONFIG

	config_get localip "$section" localip
	[ -n "$localip" ] && echo "localip  $localip" >> $CONFIG
	return 0
}

start_pptpd() {
	[ -f $DEFAULT ] && . $DEFAULT
	mkdir -p $RUN_D
	for m in arc4 sha1_generic slhc crc-ccitt ppp_generic ppp_async
ppp_mppe; do
		insmod $m >/dev/null 2>&1
	done
	ln -sfn $CHAP_SECRETS /etc/ppp/chap-secrets
	service_start $BIN $OPTIONS -c $CONFIG
}

start() {
	config_load pptpd
	setup_config pptpd || return
	config_foreach setup_login login
	start_pptpd
}

stop() {
	service_stop $BIN
	rm -rf $CHAP_SECRETS $CONFIG /etc/ppp/chap-secrets
}
```

设置防火墙向 WAN 口开放 VPN 连接

向 /etc/config/firewall 中增加如下规则。

```
config rule
      option target 'ACCEPT'
      option _name 'pptp'
      option src 'wan'
      option proto 'tcp'
      option dest_port '1723'

config rule
      option target 'ACCEPT'
      option _name 'gre'
      option src 'wan'
      option proto '47'
```

使防火墙修改生效。

```
root@OpenWrt:~# /etc/init.d/firewall reload
```

PPTP 配置账户

/etc/config/pptpd 配置文件内容如下。

```
config service 'pptpd'
        option 'enabled' '1'
        option 'localip' '172.16.1.1'
        option 'remoteip' '172.16.1.20-30'

config 'login'
        option 'username' 'hoowa'
        option 'password' '12345678'
        option 'remoteip' '172.16.1.20'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show pptpd
pptpd.pptpd=service
pptpd.pptpd.enabled=1
pptpd.pptpd.localip=172.16.1.1
pptpd.pptpd.remoteip=172.16.1.20-30
pptpd.@login[0]=login
pptpd.@login[0].username=hoowa
pptpd.@login[0].password=12345678
pptpd.@login[0].remoteip=172.16.1.20
```

pptpd.pptpd 选项的参数说明如表 16-1 所示。

表 16-1　pptpd.pptpd 选项参数说明

选项	说明	可选值及说明	必填
enabled	是否启动服务	0：禁用 1：启用	是
localip	本设备的 IP 地址	IP 地址，字符串	是
remoteip	客户端设备的 IP 地址范围	起始 – 结束 IP 地址，例如 172.16.1.20–30 表示 20~30 的全部 IP 地址	是

pptpd.@login 匿名选项允许多个存在，用于记录 pptp 的账号和密码，参数说明如表 16-2 所示。

表 16-2　pptpd.@login 匿名选项参数说明

选项	说明	可选值及说明	必填
username	账号名称	字符串	是
password	密码	字符串	是
remoteip	这个账号的 IP 地址	字符串	是

配置文件举例如下。

```
config service 'pptpd'
        option 'enabled' '1'
        option 'localip' '172.16.1.1'
        option 'remoteip' '172.16.1.20-30'

config 'login'
        option 'username' 'hoowa'
        option 'password' '12345678'
        option 'remoteip' '172.16.1.20'
```

启动 PPTP

设置启动服务。

```
root@OpenWrt:~# /etc/init.d/pptpd enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/pptpd start
```

客户端连接

客户端使用 Windows 自带的 VPN 拨号功能，填写上 IP 地址、账号、密码，即可连接进入系统，从而访问内部网络。

16.1.2 PPTP 协议 VPN 客户端

智能路由器支持拨号到 VPN 服务器，可以让智能路由器作为 VPN 的一个节点为环境中的网络设备提供私有网络服务。

安装 PPTP 客户端

安装软件包。

```
root@OpenWrt:~# opkg update
root@OpenWrt:~# opkg install ppp-mod-pptp
```

安装完成后要重启系统使驱动生效。

```
root@OpenWrt:~# reboot
```

拨号配置

PPTP 客户端拨号配置采用的是标准 /etc/config/network 主配置文件。

/etc/config/network 配置文件节点内容如下。

```
…...
config 'interface' 'vpn'
        option 'ifname'    'pptp-vpn'
        option 'proto'     'pptp'
        option 'username'  'hoowa'
        option 'password'  '123321'
        option 'server'    '60.216.189.170'
        option 'buffering'   '1'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show network.vpn
network.vpn=interface
network.vpn.ifname=pptp-vpn
network.vpn.proto=pptp
network.vpn.username=hoowa
network.vpn.password=123321
network.vpn.server=60.216.189.170
network.vpn.buffering=1
```

network.vpn 选项的参数说明如表 16-3 所示。

表 16-3　network.vpn 选项参数说明

选项	说明	可选值及说明	必填
ifname	VPN 服务的设备名称	pptp-vpn	是
proto	所采用协议	pptp	是
username	账号	字符串	是
password	密码	字符串	是
buffering	缓冲	0：禁用 1：启用	

配置文件举例如下。

```
network.vpn=interface
network.vpn.ifname=pptp-vpn
network.vpn.proto=pptp
network.vpn.username=hoowa
network.vpn.password=123456
network.vpn.server=60.xx.xx.xx
network.vpn.buffering=1
```


启动连接

作为 network 配置下的一个设备配置，在系统重新启动后会自动尝试连接。如果要手动测试连接，可以使用下面这个语法。

```
root@OpenWrt:~# /etc/init.d/network reload
```

检测是否连接成功，有结果就表示成功。

```
root@OpenWrt:~# ifconfig pptp-vpn
pptp-vpn  Link encap:Point-to-Point Protocol
          inet addr:192.168.122.46  P-t-P:192.168.122.1  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:314 (314.0 B)  TX bytes:410 (410.0 B)
```

查看内核日志了解连接过程。

```
root@OpenWrt:~# logread|grep pptp
```

允许局域网通过 VPN 上网

默认的 VPN 拨号后，本地系统可以通过 VPN 连接外网，但局域网下的计算机、手机等设备无法使用，这个时候要修改防火墙配置，将 VPN 增加到 WAN 区域后才可以生效。

```
root@OpenWrt:~# uci show -X firewall|grep name=wan
firewall.cfg06dc81.name=wan
root@OpenWrt:~# uci show firewall.cfg06dc81.network
firewall.cfg06dc81.network=wan wan6
```

修改 WAN 区域范围，增加 VPN。

```
root@OpenWrt:~# uci set firewall.cfg06dc81.network='wan wan6 wwan vpn'
root@OpenWrt:~# uci commit
```

使修改生效。

```
root@OpenWrt:~# /etc/init.d/firewall reload
```

16.2 USB 摄像头

16.2.1 USB 摄像头准备

1. 摄像头监控必备条件

- 拥有一个 USB 摄像头。
- 摄像头最好支持 MJPEG 编码，采用 MJPEG 编码，摄像头的 CPU 占用率更低。
  - 截至发稿之日，售价低于 50 元的摄像头一般不具备 MJPEG 编码功能。
  - 如果不支持 MJPEG 编码功能，10 元以上的普通摄像头一般支持 YUV 方式，这种方式 CPU 占用率在 80% 以上，但是可以满足基本功能要求。

2. 选择合适的 USB 摄像头

USB 摄像头的输出格式通常有 RGB、YUV、MJPEG 等，YUV 是未压缩的图片格式，采集的数据需要 CPU 进行大量压缩运算，导致 CPU 占用率过高，而 MJPEG 格式是在摄像头中进行了压缩处理。

可以用 Windows 下的 AMCap 软件，或根据摄像头价格及向经销商索取信息得知是否支持 MJPEG，从 AMCap 选项里面的视频捕捉接口的压缩比选择控件中可以看出来是否支持 MJPEG，没有就是不支持了。

3. 安装 USB 摄像头驱动

保持 MR808 开发板外网连接正常，然后安装所需的依赖软件包。

```
root@OpenWRT:~#opkg update
root@OpenWRT:~#opkg install kmod-usb2
root@OpenWrt:~# opkg install kmod-usb-ohci
root@OpenWrt:~# opkg install kmod-input-core
root@OpenWrt:~# opkg install kmod-video-core
root@OpenWrt:~# opkg install kmod-video-gspca-core
root@OpenWrt:~# opkg install kmod-video-videobuf2
root@OpenWrt:~# opkg install kmod-video-uvc
```

重新启动板子使驱动生效。

```
root@OpenWrt:~# reboot
```

插入摄像头硬件，同时检测是否已经驱动上（显示出 /dev/video0 表示已驱动上）。

```
root@OpenWrt:~# ls /dev/video0
/dev/video0
```

如果摄像头过于老旧，可能安装上之后会有错误信息提示：无法加载，请更换摄像头。

16.2.2 MJPG-streamer

MJPG-streamer 是一款在 Linux 下鼎鼎大名的 HTTP 方式监控软件，它支持通过 HTTP UVC 方式访问兼容的摄像头，这种方式可以让手机、计算机等设备直接观察摄像头图像内容。

安装 MJPG-streamer

```
root@OpenWrt:~# opkg install libpthread
root@OpenWrt:~# opkg install libjpeg
root@OpenWrt:~# opkg install mjpg-streamer
```

安装过程中会提示很多信息，也会有警告，安装好之后重启就会正常。

```
root@OpenWrt:~# reboot
```

配置 MJPG-streamer

/etc/config/mjpg-streamer 配置文件内容如下。

```
config mjpg-streamer 'core'
        option enabled '1'
        option input 'uvc'
        option output 'http'
        option device '/dev/video0'
```

```
        option resolution '320x240'
        option fps '5'
        option www '/www/webcam'
        option port '8080'
        option username 'openwrt'
        option password 'openwrt'
```

UCI 查看方法如下。

```
root@OpenWrt:~# uci show mjpg-streamer
mjpg-streamer.core=mjpg-streamer
mjpg-streamer.core.enabled=1
mjpg-streamer.core.input=uvc
mjpg-streamer.core.output=http
mjpg-streamer.core.device=/dev/video0
mjpg-streamer.core.resolution=640x480
mjpg-streamer.core.fps=5
mjpg-streamer.core.www=/www/webcam
mjpg-streamer.core.port=8080
mjpg-streamer.core.username=openwrt
mjpg-streamer.core.password=openwrt
```

mjpg-streamer.core 选项的参数说明如表 16-4 所示。

表 16-4 mjpg-streamer.core 选项参数说明

选项	说明	可选值及说明	必填
enable	是否启用该配置，默认为 0	0：不启用 1：启用	是
input	输入驱动类型	uvc：USB 摄像头 file：视频文件	是
output	输出驱动类型	http：http 视频流 file：视频文件	是
device	输入设备，默认为 /dev/video0	/dev/videox，x 为摄像头编号	否
resolution	分辨率，根据 CPU 能力和摄像头能力而定	320 像素 ×240 像素或 640 像素 ×480 像素	否
fps	帧数	1~30，过高的话会卡顿	否
www	http 站点的保存路径	可以将 MJPG-streamer 官网源代码中的 www 目录复制过来，大家可以看见更加美观的页面	是
username	用户名		否
password	密码		否

默认情况下，参数不用修改，设置为 enable 即可启动。

```
root@OpenWrt:~# uci set mjpg-streamer.core.enabled=1
root@OpenWrt:~# uci commit mjpg-streamer
```

启动服务

设置开机自动启动服务。

```
root@OpenWrt:~# /etc/init.d/mjpg-streamer enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/ mjpg-streamer start
```

16.2.3 浏览器访问

打开浏览器访问 http://192.168.1.1:8080，系统会提示输入设置的账号和密码，输入完成后就可以看到内容了。

将默认的 MJPEG 解码修改为 YUV 解码

如果看不到画面，可能是摄像头不支持 MJPEG 格式，需要以 YUV 方式启动。

编辑文件 /etc/init.d/mjpg-streamer。

```
[ -n "$fps" ] && input_arg="${input_arg} --fps $fps "
```

将上面这句修改为：

```
[ -n "$fps" ] && input_arg="${input_arg} --fps $fps --y"
```

完成后重新启动 MJPG-streamer 服务。

```
root@OpenWrt:~# /etc/init.d/mjpg-streamer restart
```

再次刷新浏览器即可看到图像。

设置外网访问

默认情况下，8080 端口在外网是无法访问的，需要修改默认的防火墙策略，以实现支持。

通过 Vi 或 UCI 方式增加一个外网访问策略（参考第 10 章）。

```
config camera
        option src          wan
        option proto        tcp
        option dest_port    8080
        option target       ACCEPT
```

重新启动防火墙，加载新的策略。

```
root@OpenWrt:~# /etc/init.d/firewall restart
```

16.3 迅雷远程下载

在公司上班发现一个“大片”，想要把它直接下载到家里的存储设备中，回家用 DLNA 就可以直接欣赏，这是不是非常好呢？但是如何做到呢？下面就介绍给大家一个超级静音、环保、节能的下载神器：基于 MR808 开发板的迅雷远程下载工具。

16.3.1 迅雷远程下载条件

迅雷远程下载是由迅雷公司发布的一套嵌入式软件实现的，该软件名为 Xware。在本系统中已捆绑该软件的 Xware1.0.31_mipsel_32_uclibc 版本（基于 MIPSEL 指令集 CPU、采用了 uclibc 库的系统、32 位，正符合本系统软硬件要求）。其他版本可以访问迅雷官网获取帮助。

Tips

Xware 官网：http://luyou.xunlei.com

下载必备条件

- SATA 接口的硬盘、U 盘或者移动硬盘。
- 参考第 12 章配置好存储设备。
- 参考第 13 章配置好共享中心。
- 参考第 14 章配置好家庭媒体中心。

安装迅雷远程下载

保持 MR808 开发板连接外网正常，然后安装所需的依赖软件包。

```
root@OpenWRT:~#opkg update
root@OpenWRT:~#opkg install libpthread
root@OpenWRT:~#opkg install xware
```

本地下载路径

创建一个本地路径，用来保存远程下载下来的文件，路径可以是 DLNA 的路径，也可以是网上邻居可以访问到的一个路径。

注意！如果是 DLNA 的路径，就不用创建本地下载路径了，下载的时候直接选择即可。

```
root@OpenWrt:/# mkdir -p /pub/TDDOWNLOAD
root@OpenWrt:/# chmod -R 777 /pub/TDDOWNLOAD/
```

初始化启动

需要注意的是第一次启动成功之后，会生成激活码。

启动迅雷远程下载。

```
root@OpenWrt:~# /etc/init.d/thunder start
root@OpenWrt:~# initing...
try stopping xunlei service first...
killall: ETMDaemon: no process killed
killall: EmbedThunderManager: no process killed
killall: vod_httpserver: no process killed
setting xunlei runtime env...
port: 9000 is usable.

YOUR CONTROL PORT IS: 9000

starting xunlei service...
etm path: /usr/local/bin
execv: /usr/local/bin/lib/ETMDaemon.

getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)
wget: can't connect to remote host (127.0.0.1): Connection refused
xunlei service is not ready now.

try again...(has tried 1 time(s)).
getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)
wget: can't connect to remote host (127.0.0.1): Connection refused
xunlei service is not ready now.

try again...(has tried 2 time(s)).
getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)

THE ACTIVE CODE IS: wxcugq

go to http://yuancheng.xunlei.com, bind your device with the active code.
```

```
finished.
```

屏幕中显示的“THE ACTIVE CODE IS”后面跟随的数据为当前这台机器的激活码。通过浏览器访问 http://192.168.1.1:9000/getsysinfo 也可以获得激活码信息。获得激活信息如图 16-1 所示。

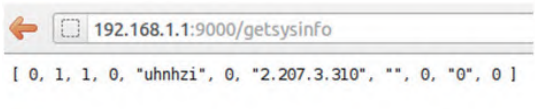


图 16-1 获得激活码信息

设置启动服务。

```
root@OpenWrt:~# /etc/init.d/thunder enable
```

开启迅雷远程下载

1 进入迅雷远程下载的官网 http://yuancheng.xunlei.com。使用你所申请的迅雷账号（不会不知道迅雷账号是什么吧？）登录。登录成功后会显示激活页面。

2 开启远程下载，必须先绑定要下载到的设备，有两种方法，如图 16-2 所示。在“输入激活码”处填入刚才安装第一次启动后获得的那个激活码，然后单击“绑定”。

图 16-2 绑定设备方法

3 成功后将进入下载界面，该界面就可以帮你远程控制 MR808 开发板去下载文件啦！

16.3.2 迅雷远程下载

在设备绑定完成后，会自动进入迅雷远程下载界面，也可以通过该界面获取迅雷远程下载客户端，这样可以通过手机、计算机等一切手段远程控制家里的设备进行下载。迅雷远程下载界面如图 16-3 所示。

图 16-3 迅雷远程下载界面

迅雷系统设置

1 由图 16-3 界面可以看到显示的设备名字为“XULEI_MIP...”，太丑陋了，这样不好看，单击文字右侧的铅笔可以修改设备名字，修改界面如图 16-4 所示，修改设备名字成功界面如图 16-5 所示。

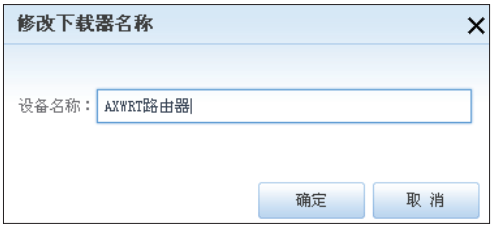


图 16-4 修改设备名字界面

图 16-5 修改设备名字成功界面

2 单击扳手图标可以设置下载参数，使用起来就和计算机上的迅雷一样。请注意将指定保存地址修改为 /pub/TDDOWNLOAD/，也可以修改为 DLNA 保存地址，否则默认会在 Linux 根目录下创建路径（磁盘中的 C：可以忽略掉，那个没用）。下载参数设置如图 16-6 所示。

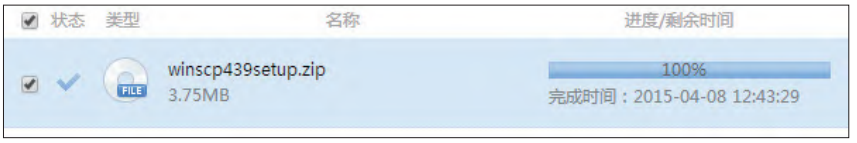
图 16-6 下载参数设置界面

迅雷远程下载

单击图 16-3 中的“新建”图标，弹出新建对话框，输入要下载的文件，单击“确认”即可创建下载任务！图 16-7 ~图 16-9 所示为举例说明如何新建下载任务。

图 16-7 新建远程下载任务

图 16-8　下载任务进度显示



■ 图 16-9　下载完成界面

在设备上查看下载文件。

```
root@OpenWrt:~# ls -l /pub/TDDOWNLOAD/
-rwxrwxrwx    1 root     root       3931201 Apr  8 04:43 winscp439setup.zip
```

16.4　WiFiDog 广告路由器

免费 Wi-Fi 用过吗？咖啡厅、机场、饭店到处都有，想必大家都体验过吧？本章将带领大家建立一个简易版的广告路由器。目前在国内，市面上所看到的广告路由器大都是基于 WiFiDog 技术进行改进的，或直接使用 WiFiDog 技术。

由于标准 WiFiDog 我们看不到认证过程，为了保证效果，本书另外封装了一个 WiFiDog 软件包，名为 WiFiDog_auth，其中包含一个基本的 demo，用于学习实践。WiFiDog 广告路由体验过程包括下面两个方面。

- WiFiDog 功能服务。
- WiFiDog 验证界面。

16.4.1　WiFiDog 功能服务

WiFiDog 原理

图 16-10 为从 WiFiDog 官网获得的 WiFiDog 工作流程图，结合这个工作流程图，给大家简单介绍一下 WiFiDog 的工作原理。WiFiDog 由 Gateway 和 Auth Server 两部分组成，启动 WiFiDog 时，Gateway 将在路由器中通过 iptables 追加一大堆规则，将 80 端口的网络数据包转发给自己的监听端口。一旦发现某一个请求还没有经过验证的话，就自动跳转到 Auth Server，要求用户按照预先设定的方式进行验证。验证成功后跳转到欢迎页面。从图 16-10 中我们发现，可以在登录页和欢迎页中植入广告。

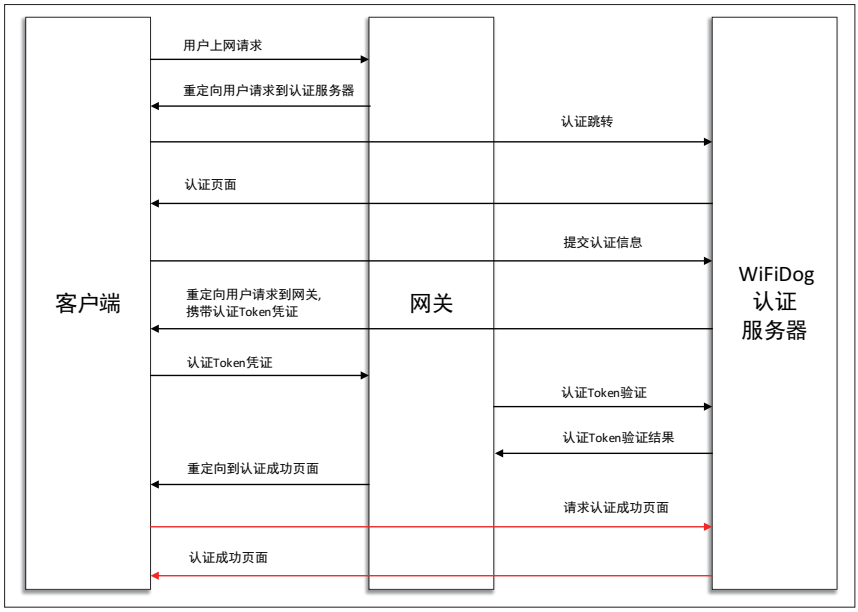


图 16-10 WiFiDog 工作流程图

安装 WiFiDog

保持 MR808 开发板外网连接正常，然后安装所需的依赖软件包。

```
root@OpenWRT:~# opkg update
root@OpenWrt:~# opkg install kmod-ipt-extra kmod-ipt-ipopt kmod-ipt-
nat-extra iptables-mod-extra
root@OpenWrt:~# opkg install iptables-mod-ipopt iptables-mod-nat-extra
root@OpenWrt:~# opkg install wifidog_auth
```

重新启动开发板，使驱动生效。

```
root@OpenWrt:~# reboot
```

以 kmod 和 iptables 开头的安装包，是可以让 WiFiDog 进行数据包转发的核心组件，大家有兴趣的话，可以深入学习 iptables。

配置 WiFiDog

/etc/wifidog.conf 配置文件内容如下。

```
GatewayID default
ExternalInterface eth0.2
GatewayInterface br-lan
```

```
AuthServer {
     Hostname 192.168.1.1
     SSLAvailable no
     Path /wifidog/pages/auth/
}
CheckInterval 60
# The timeout will be INTERVAL * TIMEOUT
ClientTimeout 5
# Parameter: TrustedMACList
# Comma separated list of MAC addresses who are allowed to pass
# through without authentication
#TrustedMACList 00:00:DE:AD:BE:AF,00:00:C0:1D:F0:0D
FirewallRuleSet validating-users {
     FirewallRule allow to 0.0.0.0/0
}
FirewallRuleSet known-users {
     FirewallRule allow to 0.0.0.0/0
}
FirewallRuleSet unknown-users {
     FirewallRule allow udp port 53
     FirewallRule allow tcp port 53
     FirewallRule allow udp port 67
     FirewallRule allow tcp port 67
}
FirewallRuleSet locked-users {
     FirewallRule block to 0.0.0.0/0
}
```


/etc/wifidog.conf 选项的参数说明如表 16-5 所示。

表 16-5　/etc/wifidog.conf 选项参数说明

选项	说明	可选值及说明	必填
GatewayID	广告路由器的唯一识别标志	任意值	是
ExternalInterface	WAN 网络对应的接口	eth0.2	是
GatewayInterface	LAN 网络的接口	br-lan	是
AuthServer Hostname	Auth Server 的服务器 IP 地址	由于在 MR808 开发板上搭建了一个简易的 Auth Server，因此为 192.168.1.1	是
AuthServer SSLAvailable	是否采用 Https 访问	yes/ no	否
AuthServer Path	Auth Server 的根目录	/wifidog/pages/auth/	是

续表

选项	说明	可选值及说明	必填
CheckInterval	检查超时的频率	默认值为 60	是
ClientTimeout	客户端超时时间，当客户端在 CheckInterval* ClientTimeout 秒之内没有任何请求的话，将自动结束会话	默认值为 5	是
TrustedMACList	受信任的客户端MAC地址列表，也就是这些客户端不需要验证就可以访问 WAN 网络		否

以上参数默认情况下不必修改即可用于测试。

启动服务

设置开机自动启动服务。

```
root@OpenWrt:~# /etc/init.d/wifidog enable
```

手动启动一次。

```
root@OpenWrt:~# /etc/init.d/ wifidog start
```

16.4.2 WiFiDog 验证界面

使用手机或计算机连接到 MR808 开发板的无线网络上。用浏览器打开一个网站测试一下看看。是否发现已经跳转到密码输入页面了？本书做了几个简单的 PHP 页面，打包到安装包中，实现了 Auth Server。密码可以在 /www/plugins/wifidog/pages/auth 目录中找到。

本系统中的验证界面依赖 Web 服务器和 PHP 语言才能正常工作，所以要完成 WiFiDog 广告路由器，需要先学习前面的 15.1 节“Web+PHP 建站系统”。

注意，确认开发板已经安装好 Lighttpd 及 PHP 语言，具备一个 Web 系统，然后再参考下面操作流程继续操作！还要确保 WiFiDog 与 Lighttpd 都已经开启，并进入服务状态。操作流程如下所示。

（1）客户端设备连接到 MR808 开发板的 Wi-Fi 信号上。

（2）MR808 开发板的 Wi-Fi 连接完成后，自动显示验证界面（可能有的人手机不显示，打开一个浏览器后将显示）。

（3）填写密码，验证成功后就可以正常上网了。验证界面如图 16-11 所示。

图 16-11　连接网络验证界面

简易验证服务器

在 WiFiDog 的配置中我们写明了 /wifidog/pages/auth/。当访问者没有经过认证之前，就将会默认访问这个页面，通过该页面实现认证，从而进入系统界面。这里我们可以通过 ls 命令查看 /wifidog/pages/auth/ 目录的内容，实现简易验证服务器。命令如下。

```
root@OpenWrt:/# ls /www/wifidog/pages/auth/
auth            js              ping            wifidog.conf
gw_message.php  login           portal
```


增强业务

广告显示？没问题吧？会 PHP 的读者是不是可以通过修改 auth 的那几个程序，实现广告显示呢？

第17章 编译定制 OpenWrt 固件

“驿外断桥边，寂寞开无主。已是黄昏独自愁，更著风和雨。无意苦争春，一任群芳妒。零落成泥碾作尘，只有香如故。”

经过了前面 16 个章节的学习，我们已经积累了很多基础知识。从本章开始，将介绍编译、驱动、定制 OpenWrt 系统的相关内容。主要包含 AxWRT 系统是如何产生的、如何根据硬件去定制系统以及如何完成简单的内核驱动程序。知识量比较大，如果读者只想学会编译系统，那么不需要具备什么知识，但如果想把这一大部分完全吃透，需要具备一定的 C 语言基础。

17.1 OpenWrt 编译环境

17.1.1 编译与交叉编译

1. 编译

将源代码生成 CPU 可以执行的目标程序，这一过程叫做编译，就像不同的语言之间翻译一样。通过对 OpenWrt 系统的编译，可以产生运行在 MR808 开发板上的程序。

2. 交叉编译

交叉编译就是在一个平台上生成另一个平台上的可执行代码。平台有两层含义：处理器的体系结构和所运行的操作系统。对于我们而言，即是在 Windows 操作系统平台上面，安装 VirtualBox，在 VirtualBox 中安装桌面 PC 版本的 Linux，然后在 Linux 中、x86 硬件平台架构上面，编译运行于 ARM 架构（S5PV210 硬件平台）的应用程序，ARM 开发板上面的操作系统是嵌入式 Linux。

3. 宿主机与目标机

宿主机（Host）：编辑和编译程序的平台，一般是基于 x86 的 PC，通常也称为主机。通常我们在 Windows 平台下使用各种编辑器编写 Linux 代码，然后在 Linux 平台下进行编译。此时的宿主机即为 VirtualBox 中的 Linux 平台。

目标机（Target）：开发系统，通常都是非 x86 平台。主机编译得到的可执行代码在目标机上运行。对于我们来说就是基于 ARM 架构的 MR808 开发板。

4. 调试模型

嵌入式 Linux 的开发平台如图 17-1 所示。

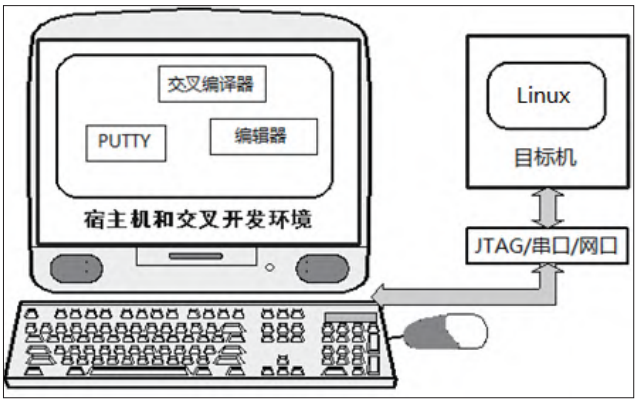


图 17-1　开发平台示意图

宿主机方面：在宿主机 Windows 操作系统中安装 VirtualBox 软件，在 VirtualBox 中安装桌面 Linux 操作系统，然后安装交叉编译器，利用交叉编译器编译适合于目标机运行的程序。

目标机方面：当电路板焊接完毕后，存储器内没有可执行程序，此时，需要宿主机通过 JTAG 下载调试器下载程序，在宿主机 JTAG 软件的辅助下，将 Bootloader（目标板引导代码）下载到目标机，Bootloader 的作用是初始化 CPU、SDRAM、串口等。当 Bootloader 下载成功后，目标机在 Bootloader 作用下就可以通过网口，反复多次快速地下载 Linux 内核及文件系统映像到目标机中调试运行。同时 Bootloader 可以通过串口与宿主机进行通信、输出调试信息等，目标机通常使用的串口终端软件有 PUTTY、SecureCRT 等。

5. 为何要交叉编译?

主要原因有以下几点。

- 目标机上不允许或不能够安装所需要的编译环境。
- 目标机上的资源贫乏，无法运行所需要编译程序。
- 目标机还没有建立，无操作系统及编译器，无法自己开发自己。

17.1.2　AxWRT 与 OpenWrt 的关系

在刚开始的章节中，我们已经介绍过 AxWRT 就是基于 OpenWrt 产生的一个用于学习的版本。AxWRT 在源代码层面与 OpenWrt 的 Barrier Breaker 发行版几乎完全一致，区别在于以下几点。

- 根据 MR808 开发板的接口定义，AxWRT 进行了专门处理。

- 裁剪掉了与 MR808 开发板无关的其他硬件支持。
- 由于官方 OpenWrt 不同程度地存在一些软件包的 Bug，本系统做了部分软件包的修正。
- 捆绑了所需要的 feeds 中的软件包，省去编译时需要连接网上资源的步骤。

Tips

AxWRT 源代码下载：http://pan.baidu.com/s/1i5chMD3
AxWRT 源代码的服务器下载：http://www.maker1000.com/makerouter/system/ srcpkgs.tgz

17.1.3 OpenWrt 的版本关系

OpenWrt 有很多发行版，几乎每间隔一段时间就会出现新的发行版。目前主要的发行版有以下几种。

- 开发代号 Backfire 10.03.1：2012 年发行。
- 开发代号 Attitude Adjustment 12.09：2013 年发行。
- 开发代号 Barrier Breaker 14.07：2014 年发行。
- 开发代号 Trunk：主线版本，无明确版本号，每年根据 Trunk 产生一个发行版。
- 开发代号 Chaos Calmer 15.05.1：2016 年发行。

Tips

OpenWrt 官方预编译好发行版本：http://downloads.openwrt.org/
OpenWrt 官方源代码下载：https://dev.openwrt.org/wiki/GetSource

相比较而言，Trunk、Chaos Calmer 中的内容比较新，但是存在诸多 Bug。各个发行版功能都比较稳定，适于产品开发。但是一个标准的官方源代码，不一定适合你手中的硬件，这时就需要进行裁剪定制，这个定制的过程包括参数定义及编译。如果所使用的硬件没有被 OpenWrt 官方所支持，就需要由具备嵌入式 Linux 经验的工程师来进行移植。

17.1.4 准备编译环境

在开始编译之前，需要准备一系列系统环境和软件包。在编译时我们需要使用第 4 章中创建的 CentOS 虚拟机或是独立计算机。接下来准备好你的编译系统，并且将其开机，确保网络连接畅通。

编译所需要的宿主机

- 独立计算机，安装 CentOS 操作系统或是 Ubuntu 操作系统。
- CPU 性能足够，首次编译所需要时间在 30 分钟 ~5 小时，视计算机性能而定，多核处理器效率更高，需具备 2GB 以上内存。
- 具备至少 20GB 的硬盘空间，硬盘的读写速度直接影响到编译速度，SSD 固态硬盘是最好的。

CentOS 上的 YUM

- YUM 是 CentOS 系统下的安装工具，可以安装上我们所需的全部编译用软件包。
- YUM 工具安装时会有提示是否继续，输入“y”，确认继续完成安装，如果过程中有下载失败的软件包，可以再重复执行安装，YUM 会自动找到没有完成的部分给予完成。

Development Tools 安装：Development Tools 中包含了大量编译用工具，安装时间较长，如果网络不好只能等待。

```
[root@axwrt_host ~]# yum groupinstall "Development Tools"
[root@axwrt_host ~]# yum install wget ncurses-devel zlib-devel vim-
enhanced
```

如果是 Ubuntu 系统，可以使用下面的命令安装所需编译软件包。

```
sudo apt-get install gcc g++ binutils patch bzip2 flex bison make
autoconf gettext texinfo unzip sharutils subversion libncurses5-dev
ncurses-term zlib1g-dev gawk asciidoc libz-dev
```

17.2 OpenWrt 固件生成

17.2.1 编译流程

编译流程如下。

（1）首先通过刚才下载的那些编译工具编译**交叉编译工具**。

（2）根据预设的参数和类型，使用**交叉编译工具**分别编译内核和各个软件包。

（3）将编译好的东西打包成固件文件（一般以 bin 结尾）。

（4）生成固件。

17.2.2 获取 AxWRT 源代码

在编译 OpenWrt 的过程中，不能使用 root 超级用户，因此，以下例程以账号 hoowa 进行操作，请各位自己创建一个普通账号完成后续操作。用以下两种方式获取源代码都可以。

返回当前账号主目录。

```
[hoowa@axwrt_host /]$ cd ~
[hoowa@axwrt_host ~]$
```

通过 HTTP 方式获得源代码。

```
[hoowa@axwrt_host ~]$ wget http://www.maker1000.com/makerouter/system/source.tgz
[hoowa@axwrt_host ~]$ tar zxf source.tgz
[hoowa@axwrt_host ~]$ mv source axwrt
```

17.2.3 获取软件包源代码

OpenWrt 在编译过程中，会去不同的网站下载所需要的软件包，例如内核会去 kernel.org 下载等，这个下载量很大，网络不太好时速度会很慢，本书已经做好了一个软件包列表来缩短 AxWRT 的编译时间。这部分软件包很大，容量超过 500MB，可以使用 wget 命令直接下载，也可以使用别的方式下载传到计算机中。命令如下所示。

```
[hoowa@axwrt_host ~]$ cd axwrt/
[hoowa@axwrt_host axwrt]$ wget http://www.maker1000.com/makerouter/
system/srcpkgs.tar
[hoowa@axwrt_host axwrt]$ tar xf srcpkgs.tar
[hoowa@axwrt_host axwrt]$ mv srcpkgs dl
```

请将 srcpkgs.tar 在你的计算机上保留好，以后还可以继续用，真心是节省时间啊。

17.2.4 OpenWrt 菜单选项配置

OpenWrt 菜单选择非常丰富，默认的 AxWRT 已经预设好了选项，如果只想体验一下编译，可以不做修改，如果某些菜单选项选择错了，可能导致编译失败，解决办法就是不断地尝试！

首次进入菜单选项，需要点时间，系统要收集、整理已经预设的软件包。

```
[hoowa@axwrt_host axwrt]$ make menuconfig
```

执行 make menuconfig 后，屏幕中将显示图 17-2 所示的菜单选项界面。

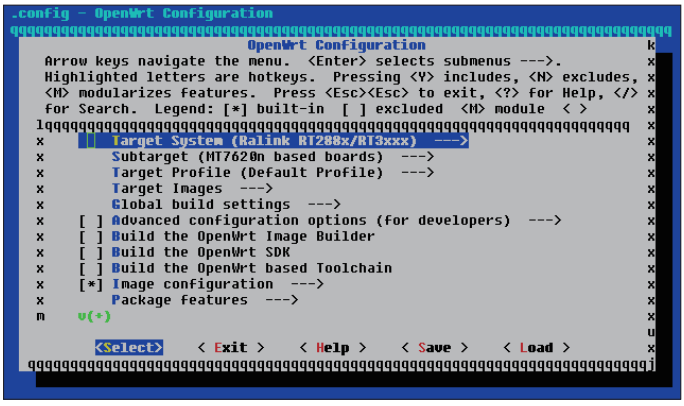


图 17-2 菜单选项界面

菜单选项的使用方法如下。

（1）通过键盘上下键可以更换选项。

（2）有“--->”符号的选项代表其有子菜单。

（3）通过按回车键进入选项或修改选项的值。

（4）通过键盘左右键可以选择菜单屏幕下面的“Select”（选择）、“Exit”（退出）、“Help”（帮助）、“Save”（保存）、“Load”（读取）选项。

（5）最终确认选择“Save”后，再选择“Exit”完成确认退出。

（6）在配置时，通常有 3 种选择方式：

① 将该功能编译进内核，即选择为“*”；

② 不将该功能编译进内核，即选择为“空”；

③ 以模块方式编译，需要时可以动态插入内核中，即选择为“M”；

按“空格”键切换 Y、N、M 选择。

图 17-2 的菜单选项包括很多项，大体上可分为表 17-1 所示的几大类。

表 17-1　菜单选项

选项	作用
Target System	目标设备芯片类型，AxWRT 系统只保留了 Ralink RT288x/RT3xxx。MR808 开发板 MT7620 芯片就是 RT 系列的升级版
Subtarget	目标子类型，一般是具体板子类型
Target Profile	针对不同路由器的版本再做的一个明确选项参数，一般按默认即可
Target Images	生成的固件文件采用何种分区格式
Global build settings	编译时的一些全局参数，这些参数与 Linux 内核或 GCC 编译器相关
Advanced configuration options	高级选项参数，用于 OpenWrt 开发人员，一般用不上
Build the OpenWrt Image Builder	除了编译固件，再编译一个固件编译环境，可以分发给团队中的其他人使用。一般是厂商不想开放自己的全部源代码，给第三方合作伙伴提供这玩意
Build the OpenWrt SDK	产生 OpenWrt SDK 环境，就是另外一个开发环境
Build the OpenWrt based Toolchain	单独编译出 OpenWrt 的交叉编译工具
Image configuration	编译好的固件所附带的参数信息
Package features	一些软件包特性
Base system	基本系统命令软件包
Administration	高级管理命令软件包
Boot Loaders	引导程序，MR808 开发板不用固件附带，所以这里无选项

续表

选项	作用
Development	开发用工具包
Firmware	不要被名字混淆了，这里的意思是，固件中是否带某些其他外围芯片的固件
Kernel modules	内核编译参数，这里的参数都来自芯片厂对内核的预设配置
Languages	编程语言软件包
Libraries	库软件包
LuCI	OpenWrt 下大名鼎鼎的 LuCI 界面包
Mail	与电子邮件相关的软件包
Multimedia	多媒体软件包
Network	与网络相关的软件包
Sound	与声音相关的软件包
Utilities	工具软件包
Video Streaming	视频流服务软件包

并不是所有的选择都要选上，实际上如果所有选项都选上，一定会出问题。一方面，编译时间可能要按天计算；另一方面，用来存放固件的 Flash 容量也不足以安装这么多东西，所以一般要根据你系统的专用性来进行选择，根据你要完成的功能、硬件电路所支持的功能有哪些来选择。

OpenWrt 的菜单选择有两种方式。

- Buildin：绑定，直接将所选择软件包绑定到固件中。
- Module：模块，所选择软件包不绑定到固件中，以模块的形式存在，在需要时使用 opkg 命令安装。

若要更换两种方式很简单，选中要更换策略的软件包，按空格键，如果显示为 [*] 则表示绑定，如果显示为 [M] 则表示模块化编译。

本书推荐第一次编译不要去调整软件包，软件包存在很多依赖关系，先成功一次，然后再去选择哪些想要，哪些不想要。

17.2.5 编译固件

使用 make 命令即可完成编译。不显示信息，直接编译，命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make
```

采用双核编译。采用多核编译可以节省至少 30% 的编译时间，但是由于软件包依赖原因可能会产生编译错误，并且不易于查找错误。参数 -j 后面跟随线程的数量，根据 CPU 的配置来填写。命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make -j 2
```

显示全部信息，采用双核编译（挂起到后台处理，但是前台显示编译信息），命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make V=99 -j 2 &
```

最好、最稳妥的还是单核编译（挂起到后台处理，但是前台显示编译信息），命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make V=99 &
```

如果编译过程中遇到错误或手动停止编译，不会出现什么问题，只要重复执行编译命令，它会从中断的地方继续编译。

首次编译 OpenWrt 时，因为要先编译交叉编译工具，因此所花费的时间是最长的，之后只要没执行过清理，再编译就会只编译改变的地方，那样编译时间就很短了。

17.2.6 清理编译

有时候我们需要清理掉已经做过的某些编译操作。清理掉已编译好的固件、软件包，命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make clean
```

清理掉已编译好的交叉编译工具、固件、软件包，命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make dirclean
```

清理掉所有内容（包含 AxWRT 定义的软件包信息，这个清理将返回 OpenWrt 的原始状态），命令如下所示。

```
[hoowa@axwrt_host axwrt]$ make distclean
```

17.2.7 编译完成

编译完成后，在 axwrt/bin 目录下就会有所编译出来的固件文件以及所有的软件包文件。将其中以 sysupgrade.bin 结尾的文件提出，使用之前章节所学习的方法就可以将

sysupgrade.bin 烧写到 MR808 开发板中，并进行测试。

```
[hoowa@axwrt_host axwrt]$ ls bin/ramips/
md5sums
openwrt-ramips-mt7620n-root.squashfs  openwrt-ramips-mt7620n-vmlinux.bin
packages
openwrt-ramips-mt7620n-mr808-squashfs-sysupgrade.bin  openwrt-ramips-
mt7620n-uImage.bin
openwrt-ramips-mt7620n-vmlinux.elf
```

17.3 OpenWrt 源码预设

17.3.1 Feeds 源

OpenWrt 的源码中仅包含了基本软件，其他软件都是从外围源上下载的软件包。这些软件包统称为 Feeds 源。

查看当前 Feeds 源的下载地址。

```
[hoowa@axwrt_host axwrt]$ cat feeds.conf.default
src-git packages https://github.com/openwrt/packages.git;for-14.07
src-git luci https://github.com/openwrt/luci.git;luci-0.12
src-git routing https://github.com/openwrt-routing/packages.
git;for-14.07
src-git telephony https://github.com/openwrt/telephony.git;for-14.07
src-git management https://github.com/openwrt-management/packages.
git;for-14.07
src-git oldpackages http://git.openwrt.org/14.07/packages.git
```

默认情况，可以通过下面的命令获得软件包，下载全部 Feeds 源并且安装到源代码中的语法如下所示。

```
./scripts/feeds update -a
./scripts/feeds install -a
```

本系统 AxWRT 中已经下载完成全部 Feeds 软件包，只是默认没有选中，如想生成一个不包含任何东西的纯净系统，要清理掉全部信息，执行下面的命令即可。

```
make distclean
```

所有来自 Feeds 的软件包都会被下载到 feeds/ 文件夹下，并且会生成相应的索引文件用于 OpenWrt 的菜单选项显示。

17.3.2 OPKG 自定义软件仓库

我们知道在 AxWRT 系统中的软件包管理工具为 OPKG，它可以在线下载软件包，在线下载的地址也是可以配置的。在 OpenWrt 菜单选项中，配置下载地址的步骤如下。

（1）选中 Image configuration，然后进入。

（2）选中 Version configuration options，然后进入。

（3）在 Release repository 处按回车键，填写 OPKG 的下载地址。

相应地，将编译好的固件中的 bin/ramips/packages 文件夹复制到所填写的 OPKG 地址中，这样就可以实现自己的软件仓库。

17.3.3 预设编译参数文件

在获得 AxWRT 或完成基本的配置进行编译之前，OpenWrt 系统已经完成了一个预设编译参数文件，这个文件包含了全部配置信息，有兴趣的读者可以看一下，开发人员会经常对两个文件进行对比操作，来检查对软件包或配置所做的修改。

当前预设编译参数文件。

```
axwrt/.config
```

上一次预设编译参数文件。

```
axwrt/.config.old
```


17.3.4 OpenWrt 文件夹的作用

OpenWrt 文件夹的作用如表 17-2 所示。

表 17-2 OpenWrt 文件夹的作用

文件夹名称	作用
bin/	编译完成后的固件文件以及软件包
build_dir/	build_dir/host/ 用于编译支持的宿主系统运行程序 build_dir/target_xxx 所有软件包的编译过程文件夹 build_dir/toolchain_xxx 交叉编译工具
config/	菜单选项的基本菜单
dl/	各类软件包下载后源代码存放位置
docs/	OpenWrt 的官方文档
feeds/	Feeds 源的软件包描述和配置
include/	编译的参数描述
package/	基本软件包描述和配置（Feeds 源也会软连接到这里）

续表

文件夹名称	作用
scripts/	用于完成编译的脚本程序
staging_dir/	存放编译后的目录结构，固件通过这里的结构生成
target/	存放各类 CPU 和芯片和电路板参数与专有驱动程序
tmp/	临时文件
toolchain/	交叉编译工具的源文件
tools/	用于辅助编译的各类工具编译参数

17.3.5 Target/Linux 下平台预设文件

系统编译时，大部分文件夹都是参与整个编译的，其中最主要的是 Target/Linux 与 package/（下章将专门讲解软件包如何制作）目录。Target/Linux 下的文件包含以下作用。

（1）描述厂商芯片标准。

（2）描述芯片类型。

（3）描述芯片的驱动。

（4）描述针对不同的板子所使用的驱动程序类型。

（5）与菜单选项中的 Target 系列选项有直接对应关系。

AxWRT 的 Target/Linux 下的内容如下。

```
[hoowa@axwrt_host axwrt]$ ls target/linux/
generic  Makefile  ramips
```

AxWRT 经过对 Target/Linux 下的内容进行精简，取消了很多本系统用不上的 CPU 类型支持。所保留的有以下两个。

- generic/：通用标准信息。
- ramips/：针对 MTK 公司的芯片。

不同芯片类型在 target/linux/xxx 下面有描述，不同种芯片的具体内容有一定区别，这是因为各芯片厂商的芯片表述不一致，这里我们仅以 ramips 文件夹为例进行说明，如表 17-3 所示。

表 17-3 target/linux/ramips

文件夹名称	作用
base-files/	用于 ramips 芯片的固件中的普通文件，这些文件都将直接打包到固件中
dts/	Linux 系统硬件驱动设备树文件，这是从 OpenWrt BB 版开始为 MTK 芯片新引进的表达方式

续表

文件夹名称	作用
files/	驱动程序或芯片汇编程序源代码
image/	针对这个芯片的固件编译参数
mt7620a/	针对 MT7620A 这款芯片的配置参数以及与其相关的板子说明
mt7620n/	针对 MT7620N 这款芯片的配置参数以及与其相关的板子说明
mt7621/	针对 MT7621 芯片的配置参数（BB 版对 7621 的支持还不成熟）
patches-3.10/	针对内核版本 3.10 的芯片相关补丁，默认 BB 版主内核
rt288x/	针对 RT288x 系列芯片的配置参数以及板子说明
rt305x/	针对 RT305x 系列芯片的配置参数以及板子说明
rt3883/	针对 RT3883 系列芯片的配置参数以及板子说明

17.4 OpenWrt 官方版编译

OpenWrt 官方版与 AxWRT 系统的编译流程上差不多，也可以使用 srcpkgs.tar 加速，但效果不明显，因为官方版的软件包一旦变化了，还需要重新下载。如果愿意折腾，也可以折腾一下，但是那玩意的 bug 有时候会让编译者卡住，这时就需要编译者耐心地去排除 bug 了。

下载 OpenWrt 源代码。

```
git clone git://git.openwrt.org/14.07/openwrt.git
```

下载和安装 Feeds 软件包。

```
cd openwrt
./scripts/feeds update -a
./scripts/feeds install -a
```

进入菜单选项进行配置。

```
make menuconfig
```

开始编译。

```
make V=99
```

17.4.1 关于编译

OpenWrt 的编译是一个需要不断实践的过程，通过反复操作逐步熟悉系统的每个细节，

关键在于建立一种自我提升的能力，OpenWrt 系统变化非常快，如果读者不能跟随这种变化而变化，那么就只能继续使用 AxWRT 系统了。要问本系统编译过多少次，应该不少于 100 次。

17.4.2 关于硬件支持

在菜单选项中，有关硬件类型的几个选项，直接影响到编译出来的东西是在哪里运行的。这些东西不是天然就有的，实际上是由 OpenWrt 开发组、芯片厂商、嵌入式工程师所贡献到项目中的，如果想要为某一款芯片增加这种支持，那么需要具备以下技能。

- 熟悉芯片。
- 得到原厂的 SDK 资料。
- 熟悉 C 语言。
- 熟悉 Makefile 编写。
- 熟悉 Linux 内核。
- 熟悉 OpenWrt 系统。

更直白地说，这件事情需要嵌入式驱动和系统级工程师来完成。那么我们这些软件出身的人可以干什么呢？简单！实现将自己定制的 OpenWrt 系统完美地运行在市面上的路由器产品上。学习完本书以上所有章节后，即可实现这个目的。

第18章 编译定制 IPK 软件包

“并刀如水，吴盐胜雪，纤手破新橙。锦幄初温，兽烟不断，相对坐调笙。”

做 IT 也不能每天只看着计算机，人生一世不易，多接受点文化和思想总是好事。本章主要介绍如何编译定制 IPK 软件包。

18.1 IPK 软件包与 OPKG 命令

OPKG 命令所安装的软件包格式为 IPK 软件包，IPK 软件包既可以在线安装，也可以本地安装，都是使用 OPKG 命令来完成的。

IPK 软件包的内容是预编译完成的二进制代码，在编译 OpenWrt 固件的时候，根据选项会编译完成软件包。软件包是否编译进固件有以下两种形式。

- 与固件捆绑：所编译出的软件包直接捆绑在固件中，不用独立安装。
- 模块软件包：所编译出的软件包不捆绑在固件中，需要通过 OPKG 命令进行安装。

18.2 IPK 软件包规范

通过 IPK 软件包制作，既可完成对特定软件包的移植，也可以完成自己的特定软件包。

在 package/ 下查看软件包的分类，包含了 Feeds 源软件包的软连接。

```
[hoowa@axwrt_host axwrt]$ ls package/
base-files  boot  devel  feeds  firmware  kernel  libs  Makefile  network  system  utils
```

在 system 下有具体的软件包，比如 UCI 软件包。

```
[hoowa@axwrt_host axwrt]$ ls package/system/uci
files  Makefile
```

我们以 UCI 软件包为例，介绍其内部主要文件。

- package/system/uci：软件包主目录。
- Makefile：软件包的菜单和编译方法，必备的，M 必须为大写。
- patches/：这个文件夹很有意思，如果它存在，编译程序会在下载和解压完源代码后，自动执行这里的脚本，对源代码打补丁。
- 根据需要创建 patches、files 等附加目录。

18.2.1 Makefile 格式与流程

UCI 软件包中，Makefile 文件的内容请参考以下内容。

```
# 引入头 rules.mk
include $(TOPDIR)/rules.mk

# 设置 PKG_* 变量
PKG_NAME:=mytest
PKG_VERSION:=1.0.1
PKG_RELEASE:=1

PKG_BUILD_DIR:=$(BUILD_DIR)/mytest-$(PKG_VERSION)
PKG_SOURCE:=mytest-$(PKG_VERSION).tar.gz
PKG_SOURCE_URL:=http://www.mytest.com/download/
PKG_MD5SUM:=9b7dc52656f5cbec846a7ba3299f73bd
PKG_CAT:=zcat

# 引入头 package.mk
include $(INCLUDE_DIR)/package.mk

# 定义 mytest 信息
define Package/mytest
  SECTION:=net
  CATEGORY:=Network
  TITLE:=My test is Test
  URL:=http://www.mytest.com/
endef

# 定义 mytest 的说明
define Package/mytest/description
 This is Test Package
endef

# 定义 mytest 安装什么
define Package/mytest/install
        $(INSTALL_DIR) $(1)/usr/sbin
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/mytest $(1)/usr/sbin/
endef

# mytest 脚本处理完毕，调用
$(eval $(call BuildPackage,mytest))
```

Makefile 的预定义如表 18-1 所示。

表 18-1 Makefile 的预定义

可定义项目	名称	必备
PKG_*	预设软件包的基本参数，如果有需要下载的就下载源代码	部分必备

续表

可定义项目	名称	必备
Package/	软件包描述信息	必备
Package/conffiles	软件包安装的配置文件全路径，每行一个。 比如：/etc/abc.conf	可选
Package/description	软件包的介绍信息	必备
Build/Prepare	预处理，在源代码被解压时处理动作	可选
Build/Configure	对执行 ./configure 的时候进行的特别处理	可选
Build/Compile	对软件包执行 make 编译的时候特别处理增加一些编译参数 比如：$（call Build/Compile/Default,FOO=bar）	可选
Build/Install	在编译完成后进行安装时的处理，如果不填写则自动执行 make install 比如：$（call Build/Install/Default,install install-foo）	可选
Build/InstallDev	需要编译完成后的库或头文件，但是又不在目标设备上使用	可选
Package/install	软件包安装什么	必备
Package/preinst	软件包安装之前处理的脚本 必须包含脚本头信息 #!/bin/sh 如果中断安装，在脚本中有 return false 即可实现	可选
Package/postinst	软件包安装之后处理的脚本 必须包含脚本头信息 #!/bin/sh	可选
Package/prerm	软件包卸载之前处理的脚本 必须包含脚本头信息 #!/bin/sh 如果中断卸载，在脚本中有 return false 即可实现	可选
Package/postrm	软件包卸载之后处理的脚本 必须包含脚本头信息 #!/bin/sh	可选

18.2.2 PKG_* 任务的作用

预设变量 PKG_* 的项目如表 18-2 所示。PKG_* 任务的作用如下。

（1）检测软件包是否被选中。

（2）软件包如果有需要下载的源代码文件，按 PKG_SOURCE_URL 进行下载。

（3）检测本地 dl/ 是否包含了源代码文件 PKG_SOURCE。

（4）如果需要校验源代码，就执行 PKG_MD5SUM。

（5）解压缩源代码。

表 18-2 预设变量 PKG_* 的项目

预设的变量	作用
PKG_NAME	软件包的名称，在菜单选项中将看见，必须与目录名称一致
PKG_VERSION	软件包版本，如果软件包的源代码是在线下载的，版本号要与服务器上的一致
PKG_RELEASE	软件包的 Makefile 版本，自己编
PKG_BUILD_DIR	软件包的编译过程目录名字，这个要与源代码解压完以后生成的目录名字一样，否则就找不到源代码了

续表

预设的变量	作用
PKG_SOURCE	软件包源代码文件的名字
PKG_SOURCE_URL	软件包源代码的下载地址，如果不填写就不下载，直接从 dl/ 下查找
PKG_MD5SUM	对下载下来的软件包的校验
PKG_CAT	指定使用哪个解压工具（zcat、bzcat、unzip）
PKG_BUILD_DEPENDS	软件包编译的时候依赖哪些其他软件包
PKG_SOURCE_PROTO	通过代码仓库获得代码 git、svn
PKG_REV	如果是通过 svn 获得代码，这表示 svn 的版本号

18.2.3 定义 Build/* 的作用

定义 Build/* 的作用如下。

（1）定义的预处理方法。

（2）配置时的处理方法。

（3）编译时的处理方法。

（4）编译安装时的处理方法。

（5）安装库文件的处理方法。

18.2.4 定义 Package/* 的作用

预设定义 Package/ 的软件包描述项目如表 18-3 所示。定义 Package/* 的作用如下。

（1）描述软件包的一些情况，用于菜单选项使用。

（2）定义软件包安装在设备上的时候安装哪些软件。

（3）安装之前的处理脚本。

（4）安装之后的处理脚本。

（5）卸载之前的处理脚本。

（6）卸载之后的处理脚本。

表 18-3 预设定义 Package/ 的软件包描述项目

项目	作用
SECTION	软件包的分类，暂时没有作用
CATEGORY	软件包所属的菜单选项中的菜单，值既可以是已经有的分类，也可以是自己的一个新分类。已经有的分类可以参考同类别软件包的 CATEGORY 写法
TITLE	软件包显示的名称
URL	软件包来源公司网站
MAINTAINER	软件包的维护者信息
DEPENDS	软件包依赖哪些软件包，被依赖的软件包会自动选中
USERID	软件包安装的时候生成账号和组。 值格式为：账号 / 组

预设变量 CONFIGURE_VARS（可选）：这个变量在执行 ./configure 配置脚本时，用于选择要附加什么参数，格式如下。

```
CONFIGURE_ARGS += \
        --libdir=/usr/lib/lighttpd \
        --sysconfdir=/etc/lighttpd \
        --enable-shared \
        --enable-static \
        --disable-rpath \
        --without-attr \
        --without-bzip2 \
```

预设变量 MAKE_FLAGS（可选）：这个变量用于定义系统默认执行 make 的时候，附加的参数有哪些，格式如下。

```
MAKE_FLAGS += \
        CFLAGS="$(TARGET_CFLAGS) $(TARGET_CPPFLAGS)" \
        LDFLAGS="$(TARGET_LDFLAGS) $(LIBS)"
```

18.2.5 IPK 软件包补充说明

1. 带有内核模块的软件包

带有内核模块的软件包稍微复杂一些，要增加一些独立选项才能实现，如下所示。

```
# 定义内核模块软件包，KCONFIG 表示在这个软件包选中的时候如何修改内核的 KCONFIG，关联的内核模块名字是什么
define KernelPackage/loop
  SUBMENU:=$(BLOCK_MENU)
  TITLE:=Loopback device support
  KCONFIG:= \
        CONFIG_BLK_DEV_LOOP \
        CONFIG_BLK_DEV_CRYPTOLOOP=n
  FILES:=$(LINUX_DIR)/drivers/block/loop.ko
  AUTOLOAD:=$(call AutoLoad,30,loop)
endef

define KernelPackage/loop/description
 Kernel module for loopback device support
endef

# 走起～ 内核模块
$(eval $(call KernelPackage,loop))
```

2. 有多菜单的软件包

有的软件包自身带有多级菜单，其实基本参数与单级菜单软件包是相同的，实现方式可

以参考 Lighttpd 软件包，这里不再复述。

3. 软件移植

软件移植到 OpenWrt 系统中，就是制作 IPK 软件包的过程，会用上本章的知识点，可尝试操作，但能否成功还不一定，因为有的软件不一定能运行在 MIPS 架构的路由器中。

18.3 简易软件包 mytest

软件包的定义方法有很多，各种组合效果不同。在这里我们首先做一个软件包 mytest，这个软件包不需要在线下载任何东西，仅复制一个文件 mytest.conf 到系统的 /etc/ 下即可。

由于这个简易软件包 mytest 不包含真正需要编译的内容，因此内容为空。避免 OpenWrt 在执行 make 的时候出现错误。

```
define Build/Compile
endef
```

18.3.1 mytest 软件包的制作

创建目录。

```
[hoowa@axwrt_host axwrt]$ mkdir package/utils/mytest package/utils/
mytest/files
```

创建一个空文件 mytest.conf。

```
[hoowa@axwrt_host axwrt]$ echo "this is test" > package/utils/mytest/
files/mytest.conf
```

创建编译文件 package/utils/mytest/Makefile，内容如下。

```
include $(TOPDIR)/rules.mk

PKG_NAME:=mytest
PKG_VERSION:=1.0.1
PKG_RELEASE:=1

include $(INCLUDE_DIR)/package.mk

define Package/mytest
  SECTION:=net
  CATEGORY:=Utilities
  TITLE:=My test is Test
  URL:=http://www.mytest.com/
endef
```

```
define Package/mytest/description
 This is Test Package
endef

define Build/Compile
endef

define Package/mytest/install
        $(INSTALL_DIR) $(1)/etc/
        $(INSTALL_BIN) ./files/mytest.conf $(1)/etc/
endef

$(eval $(call BuildPackage,mytest))
```

下面是 Makefile 中容易出错的地方，所有需要命名的地方名字必须正确，大小写是敏感的。

- 根据语法的要求，“PKG_NAME:=mytest”中间的“:=”前后不能有空格。
- 在定义的变量前面必须用 **Tab 键**跳格，不能使用空格做缩进。

  package/install 的可用宏变量如下。
- $（INSTALL_DIR）：表示创建目录，权限为 0755。
- $（INSTALL_BIN）：表示复制文件，权限为 0755。
- $（INSTALL_DATA）：表示复制文件，权限为 0644。
- $（INSTALL_CONF）：表示复制文件，权限为 0600。
- $（CP）：表示复制文件。
- $（1）：表示假设为目标系统，指路由器的系统，$（1）/etc 表示路由器上的 etc。
- ./files：这里的 ./ 表示软件包所在路径，所以 ./files 即 package/utils/mytest/files/。

18.3.2 mytest 调试

进入 OpenWrt 菜单选项，进入菜单选项 Utilities 中会自动出现 mytest，如图 18-1 所示。

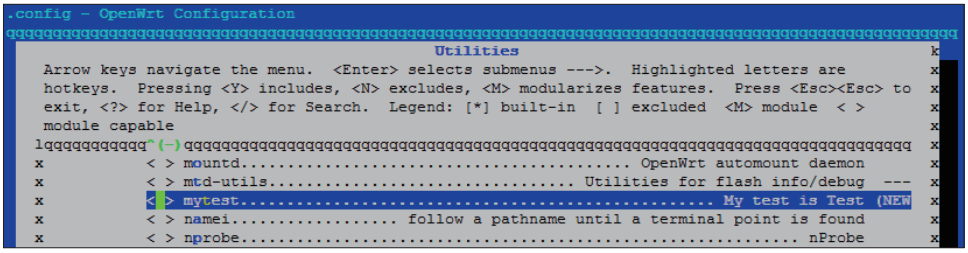


图 18-1 mytest 菜单选项

```
[hoowa@axwrt_host axwrt]$ make menuconfig
```

如果在菜单选项 Utilities 中没有发现 mytest，则表示 mytest 的脚本代码有问题，需要修改。

将 mytest 选项选择为模块 [M] 模式，然后保存退出。如果每增加一个软件包都需要重新编译固件，这不是很麻烦吗？软件包可以单独调试，语法如下所示。

```
[hoowa@axwrt_host axwrt]$ make package/mytest/{clean,compile} V=99
```

如果测试过程没有报错，说明软件包的制作应该是 OK 的，可以查看到以下结果。

```
[hoowa@axwrt_host axwrt]$ cd build_dir/target-mipsel_24kec+dsp_
uClibc-0.9.33.2/mytest-1.0.1/
[hoowa@axwrt_host mytest-1.0.1]$ find
.
./.prepared_ddb393e1aa3de2ec888800c76bd453e7
./ipkg-ramips_24kec
./ipkg-ramips_24kec/mytest
./ipkg-ramips_24kec/mytest/etc
./ipkg-ramips_24kec/mytest/etc/mytest.conf
./ipkg-ramips_24kec/mytest/CONTROL
./ipkg-ramips_24kec/mytest/CONTROL/control
./.configured_
./.built
```

在编译过程目录 build_dir/ 中，已经包含了软件包 mytest，并且可以看到生成了一个 ipkg-ramips_24kec 目录，这个目录即 IPK 软件包打包之前的目录。

18.3.3 下载测试

之前在菜单选项中，我们将 mytest 选择为 [M] 了，这时软件包已经生成了。

```
[hoowa@axwrt_host axwrt]$ find bin/|grep mytest
bin/ramips/packages/base/mytest_1.0.1-1_ramips_24kec.ipk
```

将这个软件包放到 MR808 开发板系统的 /pub 里，然后就可以执行以下命令进行安装测试了。

```
root@OpenWrt:/# cd /pub
root@OpenWrt:/pub# opkg install mytest_1.0.1-1_ramips_24kec.ipk
```

安装后看看 /etc 下是不是多了一个 mytest.conf。卸载时，再看看 /etc/ 下是不是没了 mytest.conf。

```
root@OpenWrt:/# opkg remove mytest
```

如果结合上一章介绍 OPKG 下载安装内容，就可以实现在线安装软件包了。

18.4 软件包示例 xware

xware 软件包是在前面第 16 章介绍“迅雷远程下载”时的软件包，位置在 package/utils/xware。xware 是由迅雷公司发布的不开源二进制远程下载程序，所以安装流程同样不能编译，而且没有在线下载，所有文件捆绑在软件包内部。

```
include $(TOPDIR)/rules.mk

# 定义软件包名称和版本
PKG_NAME:=xware
PKG_VERSION:=1.0.31
PKG_RELEASE:=0

# 定义软件包的 build_dir 目录
PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/package.mk

# 定义软件包的类型，软件包依赖 libpthread 这个软件包
define Package/xware
  SECTION:=utils
  CATEGORY:=Utilities
  DEPENDS:=+libpthread
  TITLE:=Xware 1.0.31
  PKGARCH:=all
endef

# 定义软件包的说明
define Package/xware/description
        xware firmware for Thunder remote download
        Xware1.0.31_mipsel_32_uclibc.zip
endef

# 告诉系统软件包公开文件
define Package/xware/conffiles
/etc/config/xware
endef

# 定义空的编译前处理，表示不做处理
define Build/Prepare
endef

# 定义空的配置，表示不配置
define Build/Configure
endef

# 定义空的编译，表示不编译
define Build/Compile
```

```
endef

# 定义软件包安装时做的事情
# 生成目录 /usr/local/bin/lib/
# 复制 ./files/usr/bin/portal 到目标系统
# 复制 ./files/usr/bin/lib/* 到目标系统
# 生成目录 /etc/init.d
# 复制可执行文件 ./init.files/thunder.init 到目标系统
define Package/xware/install
        $(INSTALL_DIR) $(1)/usr/local/bin/lib/
        $(CP) ./files/usr/bin/portal $(1)/usr/local/bin/
        $(CP) ./files/usr/bin/lib/* $(1)/usr/local/bin/lib/
        $(INSTALL_DIR) $(1)/etc/init.d
        $(INSTALL_BIN) ./init.files/thunder.init $(1)/etc/init.d/thunder
endef

# 走起!
$(eval $(call BuildPackage,xware))
```

18.5 软件包示例 shairport

shairport是我们前面介绍过的用来实现AirPlay的软件包，它的编译配置参数也很典型、简单，shairport 软件包文件说明如表 18-4 所示。

表 18-4 shairport 软件包文件说明

文件	作用
package/feeds/packages/shairport/patches/001-disable_pulseaudio.patch	补丁，编译时自动完成
package/feeds/packages/shairport/files/shairport.init	/etc/init.d/ 的启动文件
package/feeds/packages/shairport/files/shairport.config	UCI 配置文件
package/feeds/packages/shairport/Makefile	软件包编译文件

```
include $(TOPDIR)/rules.mk

# PKG_* 信息定义
PKG_NAME:=shairport
PKG_VERSION:=2014-05-08
PKG_RELEASE:=2

# 软件包是通过 git 获取的
PKG_SOURCE_PROTO:=git
PKG_SOURCE_URL:=git://github.com/abrasive/shairport.git
PKG_SOURCE_SUBDIR:=$(PKG_NAME)-$(PKG_VERSION)
```

```
PKG_SOURCE_VERSION:=078b21043cdcef6ffad7452db27f036770b23b0c
PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION)-$(PKG_SOURCE_VERSION).tar.gz
PKG_MAINTAINER:=Ted Hess <thess@kitschensync.net>

include $(INCLUDE_DIR)/package.mk

# 软件包菜单说明
define Package/shairport
  SECTION:=multimedia
  CATEGORY:=Multimedia
  DEPENDS:=+libpthread +avahi-utils +libopenssl +libavahi-client +alsa-
utils
  TITLE:=ShairPort AirPort Express emulator
endef

# 软件包说明
define Package/shairport/description
  This program emulates an AirPort Express for the purpose of streaming
  music from iTunes and compatible iPods. It implements a server for
the Apple RAOP protocol.
  ShairPort does not support AirPlay v2 (video and photo streaming).

  It supports multiple simultaneous streams, if your audio output chain
  (as detected by libao) does so.
endef

TARGET_CFLAGS += $(FPIC)

LIBS:=-lm -lcrypto -lpthread -lavahi-common -lavahi-client -lasound

# 定义编译时的一些特殊参数
MAKE_FLAGS += \
        CFLAGS="$(TARGET_CFLAGS) $(TARGET_CPPFLAGS)" \
        LDFLAGS="$(TARGET_LDFLAGS) $(LIBS)"

# 定义安装时候做的处理
define Package/shairport/install
        $(INSTALL_DIR) $(1)/usr/bin
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/shairport $(1)/usr/bin/
        $(INSTALL_DIR) $(1)/etc/init.d/
        $(INSTALL_BIN) files/shairport.init $(1)/etc/init.d/shairport
        $(INSTALL_DIR) $(1)/etc/config
        $(INSTALL_CONF) files/shairport.config $(1)/etc/config/shairport
endef

# 走起～～
$(eval $(call BuildPackage,shairport))
```

第19章 智能路由器外设应用实例

在 OpenWrt 系统中，硬件 GPIO 可以连接控制不同的用电设备，如 LED、按钮、继电器、传感器等。本章将以 LED、按钮、继电器为例，介绍智能路由器实现硬件控制。

19.1 OpenWrt 的硬件支持

19.1.1 DTS 设备树

DTS（Device Tree Source）设备树是 Linux 内核中管理设备的一种方式，通俗点讲就是告知 Linux 内核当前硬件存在何种设备。由于即使路由器芯片相同，外围电路也可能存在不同，OpenWrt 引入设备树对基本设备进行定义和配置。这些外设中，最常见和简单的就是 LED 与按钮。查看 MR808 开发板的 DTS 定义文件，源码中 target/linux/ramips/dts/MR808.dts 的内容如下所示。

```
/dts-v1/;

/include/ "mt7620n.dtsi"

/ {
   compatible = "mr808", "ralink,mt7620n-soc";
   model = "GeeQee MR808";

   palmbus@10000000 {
      gpio1: gpio@638 {
         status = "okay";
      };

      gpio2: gpio@660 {
         status = "okay";
      };

      gpio3: gpio@688 {
         status = "okay";
      };

      spi@b00 {
         status = "okay";
```

```
        m25p80@0 {
            #address-cells = <1>;
            #size-cells = <1>;
            compatible = "mx25l12805d";
            reg = <0 0>;
            linux,modalias = "m25p80", "w25q128";
            spi-max-frequency = <10000000>;

            partition@0 {
                label = "u-boot";
                reg = <0x0 0x30000>;
                read-only;
            };

            partition@30000 {
                label = "u-boot-env";
                reg = <0x30000 0x10000>;
                read-only;
            };

            factory: partition@40000 {
                label = "factory";
                reg = <0x40000 0x10000>;
                read-only;
            };

            partition@50000 {
                label = "firmware";
                reg = <0x50000 0xfb0000>;
            };
        };
    };
};

ehci@101c0000 {
    status = "okay";
};

ohci@101c1000 {
    status = "okay";
};

ethernet@10100000 {
    mtd-mac-address = <&factory 0x4>;
    ralink,port-map = "wllll";
};

wmac@10180000 {
```

```
      ralink,mtd-eeprom = <&factory 0>;
   };

   pinctrl {
      state_default: pinctrl0 {
         default {
            ralink,group = "ephy", "wled", "pa", "i2c", "wdt", "uartf",
"spi refclk";
            ralink,function = "gpio";
         };
      };
   };

   gpio-leds {
      compatible = "gpio-leds";
      indicator {
         label = "axwrt:blue:indicator";
         gpios = <&gpio1 14 0>;
      };
   };
};
```

DTS 文件大部分定义与当前芯片或电路板硬件有关，其中 MR808.dts 文件为本系统专用定义文件，DTS 文件不具备通用性，如想学习 DTS 相关的知识，可以查阅各芯片厂提供的相关资料。

19.1.2 GPIO 总线

GPIO（General Purpose Input Output，通用输入 / 输出总线）采用复杂工业总线简化出来的 I/O 扩展总线技术，市面上大部分的处理器都已支持该总线。

简单来说，GPIO 就是一条从 CPU 伸出来的线，如果将其设置为输入，则可以连接到按钮上，实现按钮按下操作，系统将接收到二进制的 1 或 0（高低电平）。如果将其设置成为输出，则可以连接到 LED 上，通过发送二进制的 1 或 0（高低电平）控制 LED 亮起或熄灭。

在 MR808 开发板上预留有 14 个 GPIO 接口，为了保证外部电路方便供电，同时也提供了 14 个 3.3V 电源接口和 14 个 GND 接口。

MR808 开发板的处理器对 GPIO 进行了分组，分组对应关系如表 19-1 所示。

表 19-1 GPIO 分组

GPIO 寄存器分组	gpio0	gpio1	gpio2	gpio3
GPIO 引脚编号	0~23	24~39	40~71	72

19.2 OpenWrt 对 LED 驱动的支持

19.2.1 LED 电路设计

LED 硬件电路设计如图 19-1 所示，电路比较简单，R1 为限流电阻，C1 为去耦电容，JP1 的 1 脚接 GND，2 脚接 3.3V 电源，3 脚接 GPIO 口，并将 GPIO 口设置为输入模式，当给 GPIO 口高电平时，LED 不亮；给低电平时，LED 点亮。

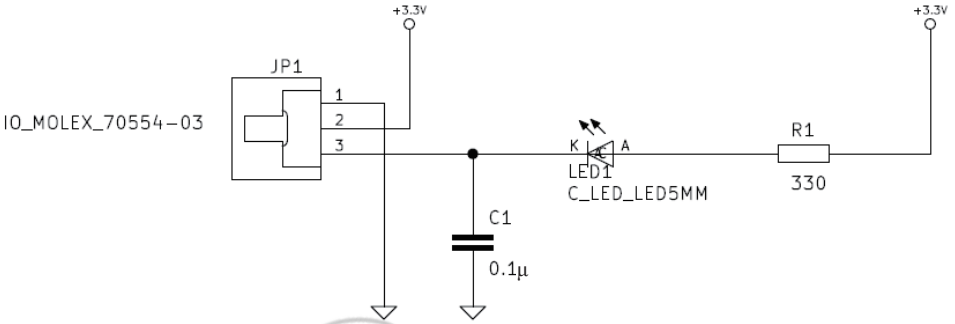


图 19-1 LED 电路原理图

19.2.2 手动测试 LED

我们可以将 MR808 开发板附赠的 LED 模块插入 MR808 开发板的 GPIO 接口中。注意：LED 模块的接口需与 MR808 开发板对应一致，否则可能导致失败。对应关系如下。

- GPIO → GPIO
- 3.3V → 3.3V
- GND → GND

这里我们将 LED 模块插入了 MR808 开发板上一组名为 WDT_RST_N/IO=17 的 GPIO 接口上。该接口编号为 17。下面临时向内核申请 GPIO 资源，来测试 LED 是否有效，如果按以下流程操作没有报错，表示操作成功。

登录 MR808 开发板系统，向系统申请 GPIO17 资源，操作命令如下。

```
root@OpenWrt:/# echo 17 > /sys/class/gpio/export
```

设置 GPIO 资源为输出模式（用以点亮 LED），操作命令如下。

```
root@OpenWrt:/# echo out > /sys/class/gpio/gpio17/direction
```

点亮 LED，操作命令如下。

```
root@OpenWrt:/# echo 0 > /sys/class/gpio/gpio17/value
```

熄灭 LED，操作命令如下。

```
root@OpenWrt:/# echo 1 > /sys/class/gpio/gpio17/value
```

如果按以上操作都没有报错，但LED没有变化，首先考虑是不是LED模块插错位置了；之后可以用万用表直接测量 GPIO 和 GND 之间的直流电压，在开关的时候 GPIO 输出的电压是高、低不同的，高为 3.3V，低为 0V。

19.2.3 将 LED 注册为系统标准 LED

如果上面测试通过，LED 可以工作正常，表示该接口设置正确，此时我们可以将其注册为系统标准的 LED。注册方法如下。

在源代码目录中编辑 target/linux/ramips/dts/MR808.dts 文件，找到以下内容。

```
        gpio-leds {
                compatible = "gpio-leds";
                indicator {
                        label = "axwrt:blue:indicator";
                        gpios = <&gpio1 14 0>;
                };
        };
```

增加内容，如下所示（注意加粗字体为增加部分）。

```
       gpio-leds {
                compatible = "gpio-leds";
                test {
                        label = "axwrt:blue:test";
                        gpios = <&gpio0 17 1>;
                };
                indicator {
                        label = "axwrt:blue:indicator";
                        gpios = <&gpio1 14 0>;
                };
        };
```

修改DTS后，需要清除掉已编译的内核，重新编译，并且将新固件刷入MR808开发板。

```
[hoowa@axwrt_host axwrt]$ make target/clean
[hoowa@axwrt_host axwrt]$ make V=99 &
```

启动系统后，可以使用 LED 标准测试方法测试亮起或熄灭，登录 MR808 开发板，测试方法如下。

```
root@OpenWrt:/# echo 1 > /sys/class/leds/axwrt\:blue\:test/brightness
root@OpenWrt:/# echo 0 > /sys/class/leds/axwrt\:blue\:test/brightness
```

19.2.4 LED 设置心跳功能

安装心跳软件包，操作命令如下。

```
root@OpenWrt:/# opkg install kmod-ledtrig-heartbeat
```

修改 /etc/config/system 配置文件，增加以下配置。

```
config 'led'
      option 'sysfs'     'axwrt:blue:test'
      option 'trigger'   'heartbeat'
```

重新启动 LED 服务，使其生效。

```
root@OpenWrt:/# /etc/init.d/led restart
```

19.2.5 将 LED 设置为 WAN 口状态

修改 /etc/config/system 配置文件，增加以下配置。

```
config 'led' 'wan'
        option 'name'          'wan'
        option 'sysfs'         'axwrt:blue:test'
        option 'trigger'       'netdev'
        option 'dev'           'eth0.2'
        option 'mode'          'link tx rx'
```

重新启动 LED 服务，使其生效，操作命令如下。

```
root@OpenWrt:/# /etc/init.d/led restart
```

19.3 OpenWrt 对按钮驱动的支持

19.3.1 按钮电路设计

按钮硬件电路设计如图 19-2 所示，JP2 的 1 脚接 GND，2 脚接 3.3V，3 脚接 GPIO 口，并设置 GPIO 口为输入模式，当按下按钮时，GPIO 口读入低电平，否则经上拉电阻 R2，将 GPIO 口置为高电平，二极管 VD1 起电路保护作用。

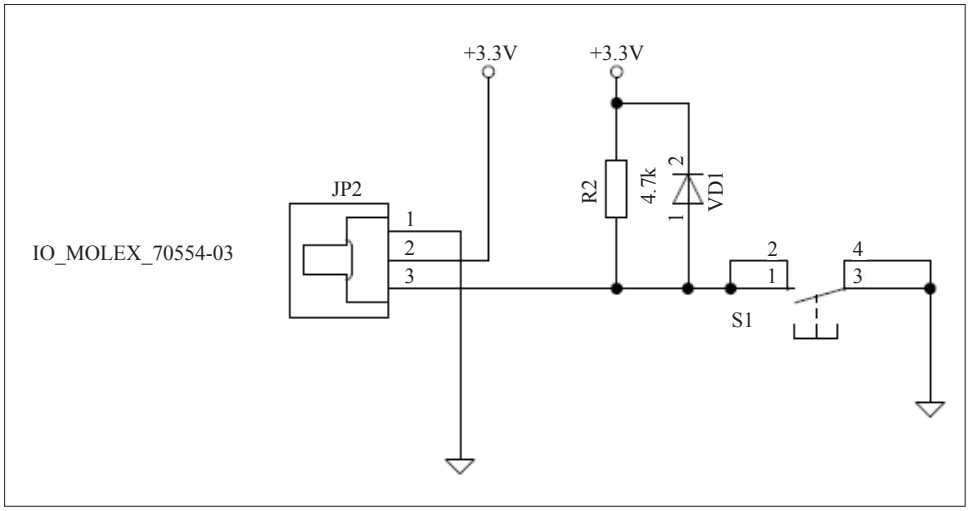


图 19-2 按钮电路原理图

19.3.2 手动测试按钮

将 MR808 开发板附赠的按钮模块插入 MR808 开发板的 GPIO 接口中。注意：按钮模块接口需与 MR808 开发板接口对应一致，否则可能导致失败，对应关系如下。

- GPIO → GPIO
- 3.3V → 3.3V
- GND → GND

我们将按钮模块插入 MR808 开发板的一组 I2C_SCLK/IO=2 的 GPIO 接口上。该接口编号为 2。下面临时向内核申请 GPIO 资源，用以测试按钮是否有效，如果按以下流程操作没有报错，则表示操作成功。

登录 MR808 开发板系统，向系统申请 GPIO 口资源，操作命令如下。

```
root@OpenWrt:/# echo 2 > /sys/class/gpio/export
```

设置 GPIO 资源为输入模式（用以接收测试按钮是否被按下），操作命令如下。

```
root@OpenWrt:/# echo in > /sys/class/gpio/gpio2/direction
```

用手按下按钮，并且查看内容，再抬起按钮，再查看内容，操作命令如下。

```
root@OpenWrt:/# cat /sys/class/gpio/gpio2/value
1
root@OpenWrt:/# cat /sys/class/gpio/gpio2/value
0
```

如果以上操作都没有报错，并且按钮按下与抬起没有变化，则考虑是不是按钮模块插错位置了；也可以用万用表直接测量 GPIO 和 GND 之间的直流电压，在按钮按下与不按下时，GPIO 输出的电压是不同的。

19.3.3 将按钮注册为系统按钮

如果上面的测试通过了，按钮工作正常，则表示该接口可以正常工作，此时我们可以将其注册为系统标准按钮。注册方法如下。

编辑 target/linux/ramips/dts/MR808.dts 文件，增加内容（加粗字体为增加部分），如下所示。

```
      pinctrl {
              state_default: pinctrl0 {
                      default {
                                  ralink,group = "ephy", "wled", "pa",
"i2c", "wdt", "uartf","spi refclk";
                                ralink,function = "gpio";
                      };
              };
      };

      gpio-keys-polled {
                compatible = "gpio-keys-polled";
                #address-cells = <1>;
                #size-cells = <0>;
                poll-interval = <20>;
                btn_1 {
                        label = "btn_1";
                        gpios = <&gpio0 2 1>;
                        linux,code = <0x101>;
                };
        };

      gpio-leds {
              compatible = "gpio-leds";
```

修改 DTS 后，需要清除掉已编译的内核，重新编译，并且将新固件刷入 MR808 开发板中。

```
[hoowa@axwrt_host axwrt]$ make target/clean
[hoowa@axwrt_host axwrt]$ make V=99 &
```

19.3.4 编写脚本检测按钮时长

编写按钮控制脚本 /etc/rc.button/BTN_1。

```
#!/bin/sh

[ "${ACTION}" = "released" ] || exit 0

. /lib/functions.sh

logger "$BUTTON pressed for $SEEN seconds"
```

设置脚本为可执行，操作命令如下。

```
root@OpenWrt:/# chmod +x /etc/rc.button/BTN_1
```

按下按钮，松开后，执行查看日志语法进行查看。

```
root@OpenWrt:/# logread|grep BTN_1
Wed Mar  2 06:30:57 2016 user.notice root: BTN_1 pressed for 0 seconds
Wed Mar  2 06:31:03 2016 user.notice root: BTN_1 pressed for 1 seconds
Wed Mar  2 06:32:50 2016 user.notice root: BTN_1 pressed for 3 seconds
```

19.4 OpenWrt 对继电器驱动的支持

19.4.1 继电器电路设计

图 19-3 所示为带光耦隔离的继电器电路设计，其中 R11 为 330Ω 的电阻，Rb 为 1kΩ 的电阻，电路中 IC3 为 TLP521 光电耦合器。当 IC3 的 2 脚为低电平时，光耦内部左侧发光二极管发光，导致 IC3 的内部右侧光电三极管导通，IC3 的 4、3 脚导通后，VT1 截止，进而继电器断开。反之，当 IC3 的 2 脚为高电平时，继电器吸合。这样，通过控制 IC3 的 2 脚电平高低进而来控制继电器的吸合与断开，从而可以实现开关量控制用电设计。图 19-4 所示为继电器 PCB 实物。本书配套开发板不提供继电器模块，可自行设计 PCB 或在网上购买。

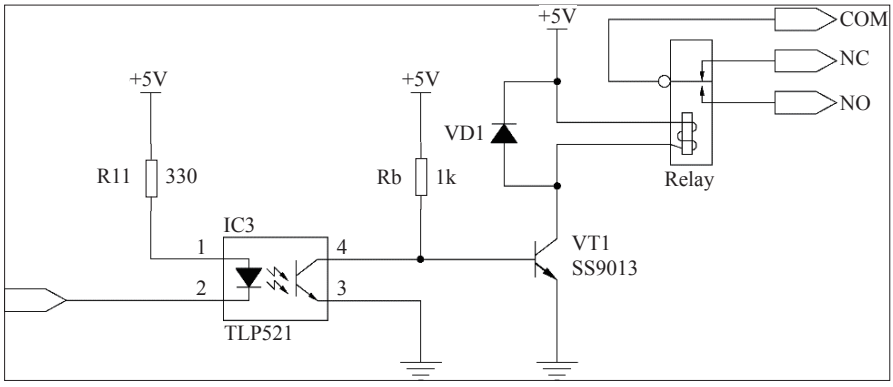


图 19-3 继电器电路原理图

图 19-4　继电器电路实物图

19.4.2　手动测试继电器

将继电器输入接口插入 MR808 开发板的 GPIO 接口中，继电器的用法与 LED 一样，与 LED 电路接法也相同，唯一不同的是，这里用的继电器供电为 5V，需要用到一个 5V 电源或能够提供 5V 电源的接口。电路连接如下。

- GPIO → GPIO
- 5V → 5V
- GND → GND

我们将继电器模块插入 MR808 开发板的 ANT_TRN/IO=19 的 GPIO 接口上，该接口编号为 19。下面临时向内核申请 GPIO 资源，用以测试继电器控制是否有效，如果按以下流程操作没有报错，则表示操作成功。

登录 MR808 开发板系统，向系统申请 GPIO 资源，操作命令如下。

```
root@OpenWrt:/# echo 19 > /sys/class/gpio/export
```

设置 GPIO 资源为输出模式（用以控制继电器开关），操作命令如下。

```
root@OpenWrt:/# echo out > /sys/class/gpio/gpio19/direction
```

继电器为开状态。

```
root@OpenWrt:/# echo 0 > /sys/class/gpio/gpio19/value
```

继电器为关状态。

```
root@OpenWrt:/# echo 1 > /sys/class/gpio/gpio19/value
```

如果以上操作都没报错，并且继电器没有动作，则要考虑是不是继电器插错位置了，或是 GPIO 口配置错了；也可以用万用表直接测量 GPIO 和 GND 之间的直流电压，测试方法与测试 LED 相同。

学做智能设备的几大误区

（1）需要具备硬件知识：学习智能设备上的软件开发和硬件调试确实需要具备硬件知识，但是所需具备的硬件知识非常非常少，少到几乎可以在几小时之内掌握，因此阅读本书的读者不需要事先具备硬件知识，但是我们希望读者具备良好的动手能力。

（2）需要懂嵌入式开发，能写驱动：完全不需要，本书中与 Linux 相关的知识跟计算机上的大部分知识都是通用的，考虑到还是有很多读者连 Linux 都没有使用过，书中也加上了 Linux 部分的内容让读者学习。

（3）需要具备多年的开发经验：也是不需要的，本书推荐读者熟悉某一种编程语言，这样便于理解各类知识，但是对编程语言以及平台没有任何要求，随便哪一种都可以，即使是 GW-BASIC 这种语言。

推荐学习者类型

本书主要从 OpenWrt 这个系统的角度来介绍，但书中内容绝不局限于路由器本身，通过阅读本书，读者可以掌握智能家居、智能玩具、工业控制等各类不同设备的玩法。本书读者对象如下：

- 计算机或电子类相关学生
- 从事 IT 行业的人员
- 智能设备或 OpenWrt 系统的发烧友
- 各智能设备厂商员工

专家推荐

我想说，很高兴看到 OpenWrt 项目在中国得到了开发者的广泛应用，希望看到你们更活跃地推动 OpenWrt 的发展，多向 OpenWrt 项目提交代码，踊跃参与在线讨论。尽管很多开发者可能存在语言障碍，但我很乐观。因为……这都不是事儿，只要一起动手做，一定能有所斩获。

——OpenWrt 社区经理 Gregers Petersen

我认识 hoowa（孙冰）超过 10 年了。从技术大牛到一个开源项目的组织者，再到科技公司的创业者，他唯一不变的还是对技术的超强理解和探索，用现在时髦的话说就是他是一名创客。创客本就应该不为商业目的去探索和付出，所以这本书的出版是再好不过的佐证。从我和他开始创业的第一天，我们就认识到未来物联网和人工智能会需要一个超强的连接器，路由器将承担这个角色，除了我们去经营一家商业化运作的公司去完成这个使命之外，还希望输出一些这个“偏门”技术领域的一些经验，让这个行业再美好一些，再单纯一些。

——GeeQee · 成都极企科技有限公司 CEO 何铮

极路由作为智能路由器的开创者，有幸与作者相识、相知，以致深度合作，一起在智能路由器、智能家居、智能办公领域为用户提供更好的产品与服务。由于智能硬件涉及嵌入式系统、硬件、云服务等多个环节，本书以 OpenWrt 系统为切入点，结合了大量实例，展现了智能路由器的强大功能，将智能路由器的能力手把手地教给读者。极路由的开发者云平台同时为读者提供实验田，期待更多的朋友加入智能硬件生态的开发队伍，感谢作者 hoowa！

——极路由 · 北京极科极客科技有限公司 董事长 王楚云


出品

封面设计：郭媛
分类建议：计算机/网络与通信/路由器
人民邮电出版社网址：www.ptpress.com.cn