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

嵌入式应用程序设计

华清远见嵌入式学院 穆煜 主编

Qianrushi
Yingyong Chengxu Sheji

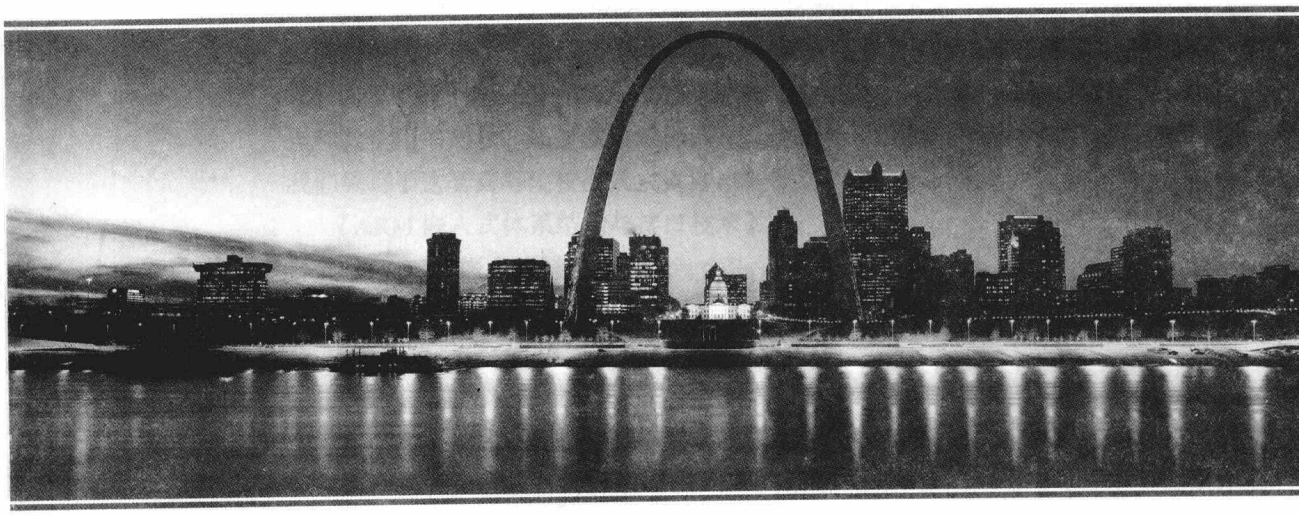
介绍实用工具
重视实际应用
全部代码示例

人民邮电出版社
POSTS & TELECOM PRESS

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

嵌入式应用程序设计

华清远见嵌入式学院 穆煜 主编



Qianrushi
Yingyong Chengxu Sheji

人民邮电出版社
北京

图书在版编目（CIP）数据

嵌入式应用程序设计 / 穆煜主编. —北京：人民邮电出版社，2009.8
高等院校嵌入式人才培养规划教材
ISBN 978-7-115-20024-2

Ⅰ. 嵌… Ⅱ. 穆… Ⅲ. Linux操作系统—程序设计—高等学校—教材 Ⅳ. TP316.89

中国版本图书馆CIP数据核字（2009）第117904号

内 容 提 要

本书结合大量实例，讲解了嵌入式 Linux 应用程序设计各个方面的基本方法，以及必要的核心概念。主要内容包括搭建嵌入式 Linux 开发环境、文件 I/O 编程、标准 I/O 编程、进程控制开发、进程间通信、多线程编程、嵌入式 Linux 网络编程、Qt 图形编程、嵌入式 Linux 设备驱动等。重视应用是贯穿全书的最大特点，本书在各章和全书结尾分别设置了在项目实践中常见和类似的应用实例。

本书可以作为高等院校嵌入式 Linux 开发课程的教材，也可供嵌入式开发人员参考。学习本书应具有 Linux C 语言编程的基本知识。

高等院校嵌入式人才培养规划教材

嵌入式应用程序设计

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

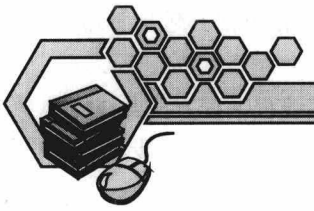
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：17.5
字数：450 千字　　2009 年 8 月第 1 版
印数：1－3 000 册　　2009 年 8 月北京第 1 次印刷

ISBN 978-7-115-20024-2

定价：29.00 元

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



前　言

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

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

《ARM 嵌入式体系结构与接口技术》

《μC/OS II 嵌入式操作系统》

《嵌入式 Linux 操作系统》

《嵌入式 Linux C 语言开发》

《嵌入式应用程序设计》

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

本书是其中之一，全书共 9 章。前 8 章是对 Linux 环境下应用开发方法的学习，各章包含相应的实验内容，第 9 章安排了一个综合实训内容。

第 1 章为搭建嵌入式 Linux 开发环境，首先介绍了嵌入式系统的基础知识、ARM 处理器的相关知识和 S3C2410 硬件平台，然后讲解了嵌入式 Linux 系统中 bootloader、内核、文件系统的构建方法。

第 2 章为文件 I/O 编程，主要讲解 Linux 系统调用、Linux 文件 I/O 系统、底层文件 I/O 操作、嵌入式 Linux 串口应用编程、标准 I/O 编程等内容。

第 3 章为嵌入式 Linux 多任务编程，主要讲解了 Linux 环境下的进程控制方法。

第 4 章为嵌入式 Linux 进程间通信，主要讲解了几种常用的进程通信方法，包括管道通信、信号通信、共享内存、消息队列等。

第 5 章为嵌入式 Linux 多线程编程，主要讲解了 Linux 环境下的多线程编程方法及注意事项。

第 6 章为嵌入式 Linux 网络编程，主要讲解了 Linux 环境下的网络编程方法，涉及网络的非阻塞访问、异步处理、多路复用等。

第 7 章为 Qt 图形编程，主要讲解嵌入式 Qt 图形编程的基础，包括嵌入式 Qt 编程环境的搭建、Qt 编程思想以及 Qt 的基本编程知识。

第 8 章为嵌入式 Linux 设备驱动编程，主要介绍 Linux 设备驱动编程基础和字符设备驱动编程的基本思路，并介绍了在 S3C2410 开发平台上编写 GPIO 驱动和按键驱动程序的基本思路。

第 9 章以一个 Qt 聊天项目设计案例，贯穿融合了前 8 章内容，提供了一种较为清晰的嵌入式应用程序开发思路。

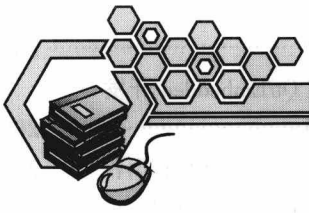
本书由穆煜主编并统校全稿。本书的完成需要感谢华清远见嵌入式学院，教材内容参考了学院与嵌入式企业需求无缝对接的、科学的专业人才培养体系。同时，嵌入式学院从业或执教多年的行业专家团队也对教材的编写工作做出了贡献，孙天泽、曾宏安、刘洪涛、赵苍明、季久峰、胡波、贾燕枫等老师在书稿的编写过程中认真阅读了所有章节，提供了大量在实际教学中积累的重要素材，对教材结构、内容提出了中肯的建议，并在后期审校工作中提供了很多帮助，在此表示衷心的感谢。

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

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

编　者

2009 年 6 月



目　录

第1章 搭建嵌入式 Linux 开发环境

本章讲解嵌入式应用开发的第一步，主要学习如何搭建嵌入式 Linux 开发的环境。本章从交叉编译环境等嵌入式开发环境的搭建开始，介绍了 Bootloader 的概念以及 U-Boot 的编译和移植的方法；然后介绍了 Linux 内核的相关知识，讲解了内核编译和移植的方法；最后还介绍了嵌入式 Linux 文件系统的构建。

本章主要内容：

- 搭建嵌入式 Linux 开发环境；
- Bootloader；
- Linux 内核与移植；
- 嵌入式文件系统构建。

1.1 搭建嵌入式 Linux 交叉开发环境

搭建开发环境是任何开发工作的基础，对于软、硬件非常丰富的嵌入式系统来说，搭建高效、稳定的环境是能否开展工作的重要因素之一。本节将介绍如何搭建一套嵌入式 Linux 开发环境。在搭建开发环境之前，有必要了解嵌入式 Linux 开发流程。因为嵌入式 Linux 开发往往涉及多个层面，这与桌面开发有很大不同。搭建一个 Linux 系统，需仔细考虑下面几点。

（1）选择嵌入式 Linux 发行版。商业的 Linux 发行版是作为产品开发维护的，经过严格的测试验证，并且可以得到厂家的技术支持。它为开发者提供了可靠的软件和完整的开发工具包。

（2）熟悉开发环境和工具。交叉开发环境是嵌入式 Linux 开发的基本模型。Linux 环境配置、GNU 工具链、测试工具甚至集成开发环境，都是开发嵌入式 Linux 的利器。

（3）熟悉 Linux 内核。因为嵌入式 Linux 开发一般需要重新定制 Linux 内核，所以熟悉内核配置、编译和移植很重要。

（4）熟悉目标板引导方式。开发板的 Bootloader 负责硬件平台的最基本的初始化，并且具备引导 Linux 内核启动的功能。由于硬件平台是专门定制的，一般需要修改编译 Bootloader。

（5）熟悉 Linux 根文件系统。高级一点的操作系统一般都有文件系统的支持，Linux 也一样离不开文件系统。系统启动必需的程序和文件都必须放在根文件系统中。Linux 系统支持的文件系统种类非常多，可以通过 Linux 内核命令行参数指定要挂接的根文件系统。

（6）理解 Linux 内存模型。Linux 是保护模式的操作系统。内核和应用程序分别运行在完全分离的虚拟地址空间，物理地址必须映像到虚拟地址才能访问。

（7）理解 Linux 调度机制和进程线程编程。Linux 调度机制影响任务的实时性，理解调度机制可以更好地运用任务优先级。此外，进程和线程编程是应用程序开发所必需的。

1.1.1 嵌入式交叉编译环境搭建

搭建交叉编译环境是嵌入式开发的第一步，也是关键的一步。不同的体系结构、不同的操作内容甚至是不同版本的内核，都会用到不同的交叉编译器。选择交叉编译器非常重要，有些交叉编译器经常会有部分的 BUG，都会导致最后的代码无法正常运行。

交叉编译器完整的安装一般涉及多个软件的安装（读者可以从 ftp://gcc.gnu.org/pub/下载），包括 binutils、gcc、glibc、glibc-linuxthreads 等软件。其中，binutils 主要用于生成一些辅助工具，如 readelf、objcopy、objdump、as、ld 等；gcc 是用来生成交叉编译器的，主要生成 arm-linux-gcc 交叉编译工具（应该说，生成此工具后已经搭建起了交叉编译环境，可以编译 Linux 内核了，但由于没有提供标准用户函数库，用户程序还无法编译）；glibc 主要是提供用户程序所使用的一些基本的函数库，glibc-linuxthreads 是线程相关函数库。这样，交叉编译环境就完全搭建起来了。

上面所述的搭建交叉编译环境比较复杂，很多步骤都涉及对硬件平台的选择。因此，现在嵌入式平台社区或厂商一般会提供在各种平台上测试通过的交叉编译器，而且也有很多把以上安装步骤全部写入脚本文件或者以发行包的形式提供，这样就大大方便了用户的使用。例如，crosstool 是美国人 Dan Kegel 开发的一套可以自动编译不同版本的交叉编译器，关于该工具的使用请参考同一系列的教材“嵌入式系统技术与设计”。

在本书中采用广泛使用的 cross-3.3.2 交叉编译器工具链，其使用非常简单。

```
$ mkdir -p /usr/local/arm  /* 这是交叉编译器安装目录*/
$ cp cross-3.3.2.bar.bz2 /usr/local/arm
$ cd /usr/local/arm
$ tar jxvf cross-3.3.2.tar.gz
```

此时在/usr/local/arm/3.3.2/bin/下已经出现了很多交叉编译工具。用户可以查看 arm 文件夹下的 VERSIONS 文件，显示如下。

```
Versions
  gcc-3.3.2
```

```
    glibc-2.3.2
    binutils-head
  Tool chain binutils configuration:
  ../binutils-head/configure …
  Tool chain glibc configuration:
  ../glibc-2.3.2/configure …
  Tool chain gcc configuration
  ../gcc-3.3.2/configure …
```

可以看到，这个交叉编译工具确实集成了 binutils、gcc、glibc 这几个软件，而每个软件也都有比较复杂的配置信息，读者可以查看 Versions 文件了解相关信息。

接下来，在环境变量 PATH 中添加路径，就可以直接使用 arm-linux-gcc 命令了。

```
$ export PATH=$PATH:/usr/local/arm/3.3.2/bin
```

把交叉开发工具链的路径添加到环境变量 PATH 中，这样可以方便地在 Bash 或者 Makefile 中使用这些工具。通常可以在环境变量的配置文件有几个。

（1）profile 类文件：用户登录时运行一次，profile 类文件包括每个用户主目录下的.profile 文件和“/etc/profile”等。用户登录时会运行用户主目录下的.profile 文件的脚本。

（2）bashrc 类文件：每当打开 bash shell 时（例如，当打开一个虚拟终端时）运行该脚本文件。bash 类文件包括每个用户主目录下的.bashrc 文件和/etc/bash.bashrc 等。

把环境变量配置的命令添加到其中一个文件中即可。

```
  $ arm-linux-gcc -v /*查看交叉编译器的版本信息*/
  Reading specs from /usr/local/arm/3.3.2/lib/gcc-lib/arm-linux/3.3.2/ specs Configured
with: ../gcc-3.3.2/configure --target=arm-linux --with-cpu=strongarm1100 --prefix=
/usr/local/arm/3.3.2 i686-pc-linux-gnu --with-headers=/work/kernel.h3900/ include
--enable-threads=pthreads --enable-shared --enable-static --enable-languages=c,
c++
  Thread model: posix
  gcc version 3.3.2
```

从上面打印的版本信息中可以看到”--prefix=/usr/local/arm/3.3.2”，这就是交叉编译器安装的路径。它是在编译前通过 prefix 选项配置的。所以，这个工具链安装的路径必须是“/usr/local/arm/3.3.2”。

1.1.2　主机交叉开发环境配置

1. 配置控制台程序

要查看目标板的输出，可以使用控制台程序。在各种操作系统上一般都有现成的控制台程序可以使用。例如，Windows 操作系统中有超级终端（Hyper Terminal）工具；Linux/Unix 操作系统有 minicom（使用 minicom 命令启动该软件）等工具。无论什么操作系统和通信工具，都可以作为串口控制台。如果在 Windows 平台上运行 Linux 虚拟机，这个串口通信软件可以任选一种。配置一个超级终端如图 1.1 所示，配置 minicom 如图 1.2（使用“minicom–s”命令进入配置界面）所示，配置参数包括串口号、每秒位数、数据位、停止位、奇偶校验、数据流控制等设置。一次配置可以保存下来，以供以后使用。

2. 配置 tftp 服务

tftp 是一个传输文件的简单协议，它基于 UDP 而实现。此协议设计的时候是为了进行小文件

传输的，因此它不具备通常的 FTP 的许多功能，它只能从文件服务器上获得或写入文件，不能列出目录，不进行认证，传输 8 位数据。

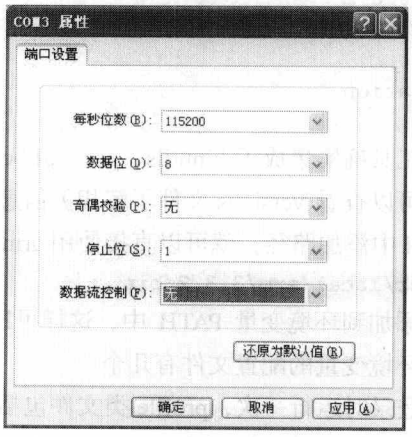


图 1.1　配置串口控制台

图 1.2　minicom 配置

tftp 分为客户端和服务器端两种。通常，首先在宿主机上开启 tftp 服务器端服务，设置好 tftp 的根目录内容（也就是供客户端下载的文件），接着，在目标板上开启 tftp 的客户端程序（tftp 客户端主要在 Bootloader 交互环境下运行，几乎所有 Bootloader 都提供该服务，用于下载操作系统内核和文件系统）。这样，把目标板和宿主机用直连线相连之后，就可以通过 tftp 协议传输可执行文件了。下面分别讲述其在 Linux 下和 Windows 下的配置方法。

（1）Linux 下 tftp 服务配置

Linux 下 tftp 的服务是由 xinetd（还有 openbsd-inetd 等其他服务）所设定的，默认情况下处于关闭状态。

首先，要修改 tftp 的配置文件，开启 tftp 服务，命令如下。

```
$ vim /etc/xinetd.d/tftp
service tftp
{
        socket_type             = dgram
        protocol                = udp
```

```
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
        disable                 = no
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

在这里，主要要将“disable=yes”改为“disable=no”，另外，从 server_args 可以看出，tftp 服务器端的默认根目录为“/tftpboot”，用户若需要可以更改为其他目录。

接下来，重启 xinetd 服务，使刚才的更改生效，命令如下。

```
$ /etc/init.d/xinetd restart
```

接着，使用命令“netstat -au”以确认 tftp 服务是否已经开启，命令如下。

```
$ netstat -au | grep tftp
Proto Recv-Q Send-Q  Local Address      Foreign Address          State
udp        0      0  *:tftp             *:*
```

这时，用户就可以把所需要的传输文件放到“/tftpboot”目录下，这样，主机上的 tftp 服务就可以建立起来了。用网络交叉线把目标板和宿主机连起来，并且将其配置成一个网段的地址，再在目标板上启动 tftp 客户端程序（注意：不同的 Bootloader 所使用的命令会有所不同，读者可以查看帮助来获得确切的命令名及格式，本书以 U-Boot 为例讲解），命令如下。

```
# tftp 0x30008000 zImage
TFTP from server 192.168.1.112; our IP address is 192.168.1.120
Filename 'zImage'.
Load address: 0x30008000
Loading:#################################################################
        #################################################################
        #################################################################
done
Bytes transferred = 881988 (d7544 hex)
```

可以看到，此处目标板使用的 IP 为“192.168.1.120”，宿主机使用的 IP 为“192.168.1.112”，下载到目标板的地址为 0x30008000，文件名为 zImage。

（2）Windows 下 tftp 服务配置

在 Windows 下配置 tftp 服务需要安装使用 tftp 服务器软件，常见的是 tftpd32，网络上有很多下载该软件的地方，读者可以自行下载。要注意的是，该软件是 tftp 的服务器端，而目标板上则是 tftp 的客户端。打开该软件，如图 1.3 所示。

接下来，用户可以在 setting 中配置服务器端的各个选项，如 IP 地址等，如图 1.4 所示。

另外，还需要在 Browse 中选择 tftp 的服务器端根目录。这时，tftpd 会提示用户重启该软件，使修改的参数生效。到此为止，tftp 的服务就配置完毕了。此时可以用直连线连接目标机和宿主机，且在目标机上开启 tftp 服务进行文件传输。

3. NFS 文件系统

NFS（Network File System）最早是由 Sun 公司提出发展起来的，其目的是让不同的计算机、不同的操作系统之间可以彼此共享文件。

NFS 可以让不同的主机通过网络将远端的 NFS 服务器共享出来的文件安装到自己的系统中，

从客户端看来，使用 NFS 的远端文件就像是使用本地文件一样。在嵌入式中使用 NFS 会使应用程序的开发变得十分方便，并且不用反复地进行烧写镜像文件。

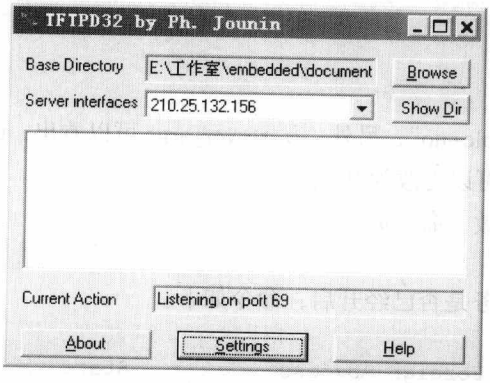


图 1.3　tftpd32 软件

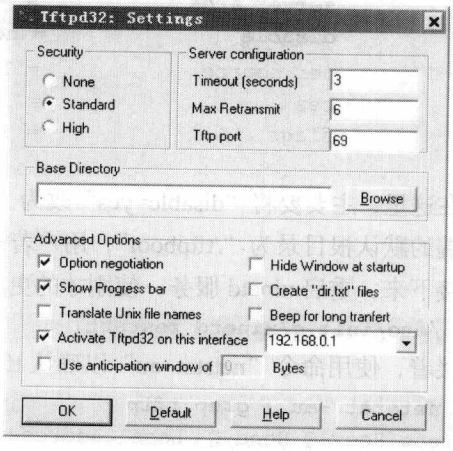


图 1.4　tftpd32 的配置界面

NFS 的使用分为服务器端和客户端，其中服务器端提供要共享的文件，而客户端则通过挂载 mount 这一动作来实现对共享文件的访问操作。在嵌入式开发中，通常 NFS 服务端在宿主机上运行，而客户端在目标板上运行。

NFS 服务器端是通过读入它的配置文件“/etc/exports”来决定所共享的文件目录的。在这个配置文件中，每一行都代表一项要共享的文件目录以及所指定的客户端对其的操作权限。客户端可以根据相应的权限，对该目录下的所有目录文件进行访问。

配置文件中每一行的格式如下。

```
[共享的目录] [客户端主机名称或 IP]([参数 1，参数 2…])
```

在这里，主机名或 IP 是可供共享的客户端主机名或 IP，若对所有的 IP 都可以访问，则可用“*”表示。这里的参数有很多种组合方式，表 1.1 列出了常见的参数。

表 1.1　　NFS 配置文件的常见参数

选　项	参 数 含 义
rw	可读写的权限
ro	只读的权限
no_root_squash	NFS 客户端分享目录使用者的权限，即如果客户端使用的是 root 用户，那么对于这个共享的目录而言，该客户端就具有 root 的权限
sync	资料同步写入到内存与硬盘中
async	资料会先暂存于内存中，而非直接写入硬盘

下面是配置文件”/etc/exports”的一个示例。

```
$ cat /etc/exports
/home/david/project    *(rw, sync, no_root_squash)
```

设定完配置文件之后，需要启动 NFS 服务和 portmap 服务，这里的 portmap 服务允许 NFS 客户端查看 NFS 服务所用的端口，在它被激活之后，就会出现一个端口号为 111 的 sun RPC（远端过程调用）的服务。这是 NFS 服务中必须实现的一项，因此，也必须把它开启，命令如下。

```
$ /etc/init.d/portmap restart
启动 portmap:                              [确定]
$ /etc/init.d/nfs restart(在 Ubuntu 中应为/etc/init.d/nfs-kernel-server)
启动 NFS 服务:                             [确定]
关掉 NFS 配额:                             [确定]
启动 NFS 守护进程:                         [确定]
启动 NFS mountd:                           [确定]
```

可以看到，在启动 NFS 服务的时候启动了 mountd 进程，它是 NFS 挂载服务，用于处理 NFS 递交过来的客户端请求。另外还会激活至少两个以上的系统守护进程，然后开始监听客户端的请求，用 dmesg 命令（或者“cat /var/log/messages”）可以看到操作是否成功。另外与 NFS 相关的还有两个命令，可以方便 NFS 的使用。

其一是 exportfs，它可以重新扫描“/etc/exports”，使用户在修改“/etc/exports”配置文件时不需要每次重启 NFS 服务，其格式如下。

```
exportfs [选项]
```

表 1.2 所示为 exportfs 的常见选项。

表 1.2　　常见选项

选　项	参 数 含 义
-a	全部挂载（或卸载）/etc/exports 中的设定文件目录
-r	重新挂载/etc/exports 中的设定文件目录
-u	卸载某一目录
-v	在 export 的时候，将共享的目录显示到屏幕上

用户若希望 NFS 服务在每次系统引导时自动开启，可使用以下命令。

```
# /sbin/chkconfig nfs on
（在 Ubuntu 中应该输入 /sbin/chkconfig nfs-kernel-server on）
```

1.2 Bootloader

Bootloader 是在操作系统运行之前执行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映像表，从而建立适当的系统软硬件环境，为最终调用操作系统内核做好准备。

对于嵌入式系统来说，Bootloader 是基于特定硬件平台来实现的。因此，几乎不可能为所有的嵌入式系统建立一个通用的 Bootloader，不同的处理器架构都有不同的 Bootloader。Bootloader 不但依赖于 CPU 的体系结构，而且依赖于嵌入式系统板级设备的配置。对于两块不同的嵌入式板而言，即使它们使用同一种处理器，要想让运行在一块板子上的 Bootloader 程序也能运行在另一块板子上，一般也都需要修改 Bootloader 的源程序。

反过来，大部分 Bootloader 仍然具有很多共性，某些 Bootloader 也能够支持多种体系结构的嵌入式系统。例如，U-Boot 就同时支持 PowerPC、ARM、MIPS 和 x86 等体系结构，支持的板子有上百种。通常，它们都能够自动从存储介质上启动，都能够引导操作系统启动，并且大部分都

可以支持串口和以太网接口。

1.2.1 Bootloader 的种类

嵌入式系统世界已经有各种各样的 Bootloader，种类划分也有多种方式。除了按照处理器体系结构不同划分以外，还有按功能复杂程度的不同来划分的。

首先区分一下 Bootloader 和 Monitor 的概念。严格来说，Bootloader 只是引导设备并且执行主程序的固件；而 Monitor 还提供了更多的命令行接口，可以进行调试、读写内存、烧写 Flash、配置环境变量等。Monitor 在嵌入式系统开发过程中可以提供很好的调试功能，开发完成以后，就完全设置成了一个 Bootloader。所以，习惯上大家把它们统称为 Bootloader。

表 1.3 所示为 Linux 的开放源码引导程序及其支持的体系结构。表中给出了 x86、ARM、PowerPC 体系结构的常用引导程序，并且注明了每一种引导程序是不是 Monitor。

表 1.3　　开放源码的 Linux 引导程序

Bootloader	Monitor	描　述	x86	ARM	PowerPC
LILO	否	Linux 磁盘引导程序	是	否	否
GRUB	否	GNU 的 LILO 替代程序	是	否	否
Loadlin	否	从 DOS 引导 Linux	是	否	否
ROLO	否	从 ROM 引导 Linux 而不需要 BIOS	是	否	否
Etherboot	否	通过以太网卡启动 Linux 系统的固件	是	否	否
LinuxBIOS	否	完全替代 BIOS 的 Linux 引导程序	是	否	否
BLOB	否	LART 等硬件平台的引导程序	否	是	否
vivi	是	主要为 S3C2410 等三星处理器引导 Linux	否	是	否
U-Boot	是	通用引导程序	是	是	是
RedBoot	是	基于 eCos 的引导程序	是	是	是

1. x86

x86 的工作站和服务器上一般使用 LILO 和 GRUB。LILO 曾经是 Linux 发行版主流的 Bootloader。不过现在几乎所有的发行版已经都使用了 GRUB，GRUB 比 LILO 有更有好的显示接口，使用配置也更加灵活方便。

在某些 x86 嵌入式单板机或者特殊设备上，会采用其他的 Bootloader，如 ROLO。这些 Bootloader 可以取代 BIOS 的功能，能够从 Flash 中直接引导 Linux 启动。现在 ROLO 支持的开发板已经并入 U-Boot，所以 U-Boot 也可以支持 x86 平台。

2. ARM

ARM 处理器的芯片商很多，所以每种芯片的开发板都有自己的 Bootloader。结果 ARM Bootloader 也变得多种多样。最早有 ARM720 处理器的开发板的固件，又有了 armboot，StrongARM 平台的 BLOB，还有 S3C2410 处理器开发板上的 vivi 等。现在 ARMBoot 已经并入了 U-Boot，所以 U-Boot 也支持 ARM/XSCALE 平台。U-Boot 已经成为 ARM 平台事实上的标准 Bootloader。

3. PowerPC

PowerPC 平台的处理器有标准的 Bootloader，就是 PPCBoot。PPCBoot 在合并 ARMBoot 等之

后，创建了 U-Boot，成为各种体系结构开发板的通用引导程序。U-Boot 仍然是 PowerPC 平台的主要 Bootloader。

4. MIPS

MIPS 公司开发的 YAMON 是标准的 Bootloader，也有许多 MIPS 芯片商为自己的开发板写了 Bootloader。现在，U-Boot 也支持 MIPS 平台。

5. SH

SH 平台的标准 Bootloader 是 sh-boot。RedBoot 在这种平台上也很好用。

6. M68K

M68K 平台没有标准的 Bootloader。RedBoot 能够支持 M68K 系列的系统。

值得说明的是 RedBoot，它几乎能够支持所有的体系结构，包括 MIPS、SH、M68K 等。RedBoot 是以 eCos 为基础，采用 GPL 许可的开源软件工程。现在由 core eCos 的开发人员维护，源码下载网站是 http://www.ecoscentric.com/snapshots。RedBoot 的文档也相当完善，有详细的使用手册《RedBoot User's Guide》。

1.2.2 U-Boot 编译与使用

最早，DENX 软件工程中心的 Wolfgang Denk 基于 8xxrom 的源码创建了 PPCBoot 工程，并且不断添加处理器的支持。后来，Sysgo Gmbh 把 PPCBoot 移植到 ARM 平台上，创建了 ARMBoot 工程。然后以 PPCBoot 工程和 ARMBoot 工程为基础，创建了 U-Boot 工程。

现在，U-Boot 已经能够支持 PowerPC、ARM、x86、MIPS 体系结构的上百种开发板，已经成为功能最多、灵活性最强并且开发最积极的开放源码 Bootloader。U-Boot 的源码包可以从 sourceforge 网站下载，还可以订阅该网站活跃的 U-Boot Users 邮件论坛，这个邮件论坛对于 U-Boot 的开发和使用都很有帮助。

U-Boot 软件包下载网站：http://sourceforge.net/project/U-Boot。

U-Boot 邮件列表网站：http://lists.sourceforge.net/lists/listinfo/U-Boot-users/。

DENX 相关的网站：http://www.denx.de。

1. U-Boot 编译

解压 U-Boot-1.3.4.tar.bz2 就可以得到全部 U-Boot 源程序。在顶层目录下有 18 个子目录，分别存放和管理不同的源程序。这些目录中所要存放的文件有其规则，可以分为 3 类。

（1）与处理器体系结构或者开发板硬件直接相关。

（2）一些通用的函数或者驱动程序。

（3）U-Boot 的应用程序、工具或者文件。

表 1.4 所示为 U-Boot 顶层目录下各级目录的存放原则。

表 1.4　　U-Boot 的源码顶层目录说明

目　录	特　性	解 释 说 明
board	平台依赖	存放电路板相关的目录文件，如 RPXlite(mpc8xx)、smdk2410(arm920t)、sc520_cdp(x86) 等目录
cpu	平台依赖	存放 CPU 相关的目录文件，如 mpc8xx、ppc4xx、arm720t、arm920t、xscale、i386 等目录

续表

目　录	特　性	解 释 说 明
lib_ppc	平台依赖	存放对 PowerPC 体系结构通用的文件，主要用于实现 PowerPC 平台通用的函数
lib_arm	平台依赖	存放对 ARM 体系结构通用的文件，主要用于实现 ARM 平台通用的函数
lib_i386	平台依赖	存放对 X86 体系结构通用的文件，主要用于实现 X86 平台通用的函数
include	通用	头文件和开发板配置文件，所有开发板的配置文件都在 configs 目录下
common	通用	通用的多功能函数实现
lib_generic	通用	通用库函数的实现
net	通用	存放网络相关程序
fs	通用	存放文件系统相关程序
post	通用	存放上电自检程序
drivers	通用	通用的设备驱动程序，主要有以太网接口的驱动
disk	通用	硬盘接口程序
rtc	通用	RTC 的驱动程序
dtt	通用	数字温度测量器或者传感器的驱动
examples	应用例程	一些独立运行的应用程序的例子，如 helloworld
tools	工具	存放制作 S-Record 或者 U-Boot 格式的镜像等工具，如 mkimage
doc	文档	开发使用文档

U-Boot 的源代码包含对几十种处理器、数百种开发板的支持。可是对于特定的开发板，配置编译过程只需要其中部分程序。这里具体以 S3C2410 处理器为例，具体分析 S3C2410 处理器和开发板所依赖的程序，以及 U-Boot 的通用函数和工具。

U-Boot 的源代码是通过 gcc 和 Makefile 组织编译的。顶层目录下的 Makefile 首先可以设置开发板的定义，然后递归地调用各级子目录下的 Makefile，最后把编译过的程序链接成 U-Boot 映像。

（1）顶层目录下的 Makefile

它负责 U-Boot 整体配置编译。按照配置的顺序阅读其中关键的几行。

每一种开发板在 Makefile 中都需要有板子配置的定义。例如，smdk2410 开发板的定义如下。

```
smdk2410_config    :    unconfig
    @./mkconfig $(@:_config=) arm arm920t smdk2410 NULL s3c24x0
```

执行配置 U-Boot 的命令 make smdk2410_config，通过./mkconfig 脚本生成 include/config.mk 的配置文件。文件内容正是根据 Makefile 对开发板的配置生成的。

```
ARCH   = arm
CPU    = arm920t
BOARD  = smdk2410
SoC    = s3c24x0
```

上面的 include/config.mk 文件定义了 ARCH、CPU、BOARD、SoC 这些变量。这样硬件平台依赖的目录文件可以根据这些定义来确定。SMDK2410 平台相关目录如下。

① board/smdk2410/。

② cpu/arm920t/。

③ cpu/arm920t/s3c24x0/。