君正®RD4730_PMP开发板

软件开发指南

版本: 1.1

日期: 2006 年 6 月

北京君正集成电路有限公司

Ingenic Semiconductor Co. Ltd

君正 **RD4730_PMP** 开发板

软件开发指南

Release history

Date	Revision	Change
Jun. 2006	1.1	Modified the description of MMC/SD, camera and busybox.
Apr. 2006	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co. Ltd

22th Floor, Building A, Cyber Tower, No.2, Zhong Guan Cun South Avenue
Haidian District, Beijing 100086, China
Tel: 86-10-82511297
Fax: 86-10-82511589
Http: //www.ingenic.cn

内容

1 概述

本文将讲述在君正 RD4730_PMP 板上开发和运行 Linux 的过程。包括在主机上建立交叉编译环境、编译 U-Boot 引导程序、编译 Linux 内核、制作根文件系统和引导和调试目标板的方法步骤。

在开发阶段，需要有一台可以提供 DHCP 服务，TFTP 服务和 NFS 服务的主机服务器。DHCP 服务主要为 RD4730_PMP 提供动态获取 IP，TFTP 服务主要为 RD4730_PMP 提供从网络下载 Linux 核心，而 NFS 服务主要为 RD4730_PMP 提供网络根文件系统。

RD4730_PMP 从 NOR FLASH 启动，引导程序为 U-Boot。U-Boot 是为嵌入式平台开发提供的开放源代码的引导程序，它提供串行口下载、以太网下载等多种下载方式，提供 NOR 闪存管理和灵活的环境变量管理功能。

RD4730_PMP 的系统核心 CE-Linux™是面向消费电子类应用的嵌入式 Linux™。它基于 Linux 2.4.20 内核，增加了抢占式内核、O(1)进程调度器、POSIX 定时器等，提高了系统的实时性。它在系统启动速度、核心尺寸和电源功耗管理支持方面也都有明显的改善。

RD4730_PMP 的根文件系统采用 BusyBox-1.1.3，支持 GLIBC 动态运行库和相关工具。

RD4730_PMP 的图形引擎为 QTOPIA/QTE，可以演示 MP3/MPEG4 媒体播放功能，JPEG 图片浏览功能，游戏以及其他等。还可以支持 Keypad 按键和触摸屏操作，电池电量显示，简单的手写输入等。

2 安装开发环境

2.1 PMP 的硬件连接

在开发阶段，您需要连接上 PMP_DEBUG 板，上面带有 JTAG 端口、Ethernet 端口以及串行端口。同时，您还需要准备一根 RS232 的串行交叉线、一条以太网线以及一个 6V 的直流电源。交叉串行线连接到主机的 COM 端口当作输出控制台，主机端 COM 端口参数设置为：115200bps，8-bit 数据，1-bit 停止位，无奇偶校验位和无硬流控。

2.2 PMP 的软件资源

RD4730_PMP 提供完全免费的软件开发工具和源码包，列表如下：

- mipseltools-jz-linux-20060308.tar.bz2：Linux 主机上的 MIPS 交叉编译工具链
- mipseltools-jz-cygwin-20060308.tar.bz2：Windows 主机上的 MIPS 交叉编译工具
- u-boot-1.1.3.tar.bz2：U-Boot 源码包
- u-boot-1.1.3-jz-yyyymmdd.patch.gz: U-Boot 补丁，'yyyymmdd'表示版本号
- celinux-040503.tar.bz2：CELinux 源码包
- celinux-jz-yyyymmdd.patch.gz：CELinux 补丁，'yyyymmdd'表示版本号
- pmp-root.tgz：PMP 根文件系统
- pmp-utils-src.tar.bz2：PMP 工具源码包

注意：请从我们的网站下载最新版本的 U-Boot 和 CELinux 补丁。

2.3 安装交叉编译工具

RD4730_PMP 采用 MIPS 交叉编译工具编译 U-Boot，Linux 核心和应用程序。我们分别提供 Linux 和 Windows 上的交叉编译工具链。建议您在 Linux 主机上进行软件开发。

在 Linux 主机上安装 MIPS 交叉编译工具的步骤如下：

首先，创建一工作目录，并把工具链包解压到该目录下，比如：

```
$ mkdir –p  /opt/crosstool
$ cd /opt/crosstool
$ tar -xjf mipseltools-jz-linux-20060308.tar.bz2
```

其次，更新 PATH 环境变量：

```
$ export PATH=/opt/crosstool/mipseltools/bin:$PATH
```

在 Windows 主机上安装 MIPS 交叉编译工具的步骤如下：

首先，安装和配置好 Cygwin 环境。请从网上下载 Cygwin 的相关软件包和资料，并参考 Cygwin 的文档进行安装。当安装好 Cygwin 后，启动 Cygwin 的 Shell 程序，并按照下面步骤进行安装：

```
$ mkdir –p /opt/crosstool
$ cd /opt/crosstool
$ tar xjf mipseltools-jz-cygwin-20060308.tar.bz2
$ export PATH=/opt/crosstool/mipseltools/bin:$PATH
```

2.4 安装 NFS 网络文件系统

在开发阶段，您需要配置一个 NFS 网络文件系统来为目标板提供 ROOT 文件系统。请把我们为您提供的根文件系统包解开到 NFS 服务的共享目录下，如”/nfsroot/pmp-root”。

```
# mkdir -p /nfsroot/pmp-root
# cd /nfsroot/pmp-root
# tar -xjf pmp-root.tar.bz2
```

设置 NFS 服务器的目的是将”/nfsroot/pmp-root”目录输出，作为目标机的根文件系统。修改配置文件/etc/exports，同时重新启动 Linux 主机上的 NFS Server 服务，如下：

```
# cat /etc/exports
/nfsroot/pmp-root    (rw,no_root_squash)
# /etc/rc.d/init.d/nfs restart
```

这时您应该可以通过网络上的任意主机挂载 NFS 文件系统，假设您的 Linux 主机 IP 为 192.168.1.20，执行下面命令挂载 NFS 根文件系统：

```
# mount –t nfs 192.168.1.20:/nfsroot/pmp-root /mnt
```

如果 mount 成功，则说明 NFS Server 服务已经正常工作。

2.5 启动 TFTP 服务

TFTP 服务主要为 U-Boot 下载 Linux 核心到目标板运行。Linux 主机上启动 TFTP 服务的过程如下：

首先，编辑/etc/xinetd.d/tftp 为如下的内容（假定使用/tftpboot 作为提供 TFTP 服务的目录）：

```
service tftp
{
   disable              =   no
   socket_type          =   dgram
   protocol             =   udp
   wait                 =   yes
   user                 =   root
   server               =   /usr/sbin/in.tftp.d
   server_args          =   -s /tftpboot
   … …
}
```

然后，运行如下命令：

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

2.6 安装 U-Boot 和 CELinux

首先，创建一工作目录：

```
# mkdir –p /home/junzheng
# cd /home/junzheng
```

然后，把 u-boot.1.1.3.tar.bz2、u-boot-1.1.3-jz-yyyymmdd.patch.gz、celinux-040503.tar.bz2、celinux-jz-yyyymmdd.patch.gz 拷贝到/home/junzheng 目录下并解开，如下：

```
# tar –xjf u-boot-1.1.3.tar.bz2
# cd u-boot-1.1.3
# zcat ../u-boot-1.1.3-jz-yyyymmdd.patch.gz | patch –p1
# cd ..
# tar –xjf celinux-040503
# cd celinux-040503
# zcat ../celinux-jz-yyyymmdd.patch.gz | patch –p1
```

2.7 安装 PMP Utilities

PMP utilities 主要包含 DPM 守护进程 dpmd 和 Camera 采集工具的源码。解开其到工作目录下并编译：

tar –xjf pmp-utils-src.tar.bz2

```
# cd pmp-utils
# cd dpmd
# make clean; make
# cd ..
# cd camera
# make clean; make
```

3 U-Boot 和 CELinux

本节将介绍编译 U-Boot 和 CELinux 的步骤，以及将 U-Boot 和 CELinux 下载到 PMP 板上运行的方法。

3.1 编译 U-Boot 和 CELinux

编译 U-Boot 的步骤如下：

```
# cd /home/junzheng/u-boot-1.1.3
# make pmp_config
# make
```

编译完成后会生成 U-Boot 的二进制映像 u-boot.bin。把 u-boot.bin 拷贝到 TFTP 服务的目录下（如/tftpboot），并根据 JDI 用户手册介绍的方法通过 JDI 烧录到目标板的 NOR FLASH 上。

注意：CPU 从 NOR FLASH 的启动地址为 0xbfc00000，请把 U-Boot 烧录到 NOR FLASH 的 0xbfc00000 地址上。

编译 CELinux 的步骤如下：

```
# make defconfig-pmp
# make xconfig
# make dep
# make uImage
```

“make uImage”编译生成 U-Boot 可以使用的 Linux 核心二进制映像 uImage，位于 celinux-040503/arch/mips/uboot/目录下。把 uImage 拷贝到 TFTP 服务的目录下（如/tftpboot），接下来就可以启动和配置 U-Boot 来下载 uImage 运行了。

U-Boot 的二进制映像烧录到 NOR FLASH 后，给 PMP 板连接好网线，串口线和电源线。在主机端启动串口终端，配置参数为：115200bps，8 个数据位，1 个停止位，无奇偶校验位。然后给 PMP 板上电，按 SW1 电源开关启动目标板，这时在串口终端可以看到 U-Boot 的输出信息。如果没有输出信息，请检查硬件跳线开关和连线是否正确。

在看到 U-Boot 输出信息后，按任意一键跳过自动启动过程进入到 U-Boot 命令界面。在 U-Boot 命令界面里运行“help”命令，可以看到 U-Boot 支持的所有命令，运行“help command”查看具体命令的格式和使用方法，比如：

```
PMP # help
PMP # help tftpboot
```

这时，您需要配置 U-Boot 从网络通过 TFTP 下载 Linux 核心运行，并挂载 NFS 网络文件系统。请参考下面的步骤来进行配置：（这里假设您的服务器 IP 地址为 192.168.1.20，并且服务器已经启动了 TFTP 和 NFS 服务；Linux 核心位于 TFTP 服务的目录/tftpboot 下，文件名为 uImage；网络文件系统路径为/nfsroot/pmp-root；PMP 的网络 IP 为 192.168.1.10，MAC 地址为 00:12:34:56:78:90）

```
PMP # setenv ipaddr 192.168.1.10
PMP # setenv serverip 192.168.1.20
PMP # setenv bootfile uImage
PMP # askenv bootcmd
bootcmd: tftpboot;bootm
PMP # setenv bootargs mem=64M console=ttyS3,115200 ip=192.168.1.10
ethaddr=00:12:34:56:78:90 nfsroot=192.168.1.20:/nfsroot/pmp-root rw
PMP # saveenv
```

如果以上步骤都正常，这时重启 PMP 板（按 SW2 开关），您就可以在串口终端看到 U-Boot 通过 TFTP 下载 uImage 然后运行 CELinux 的过程了。Linux 将通过 NFS 挂载 ROOT，成功后，在串口终端将会看到 shell 界面。这时您还可以通过 telnet 登陆到 PMP，如下：

```
# telnet 192.168.1.10
```

您还可以通过网络 TFTP 下载 uImage 到 PMP 的 NOR FLASH，然后从 NOR FLASH 运行 Linux，步骤如下：

```
PMP # setenv ipaddr 192.168.1.10
PMP # setenv serverip 192.168.1.20
PMP # erase 0xbfd00000 0xbfffffff
PMP # tftpboot 0xbfd00000 uImage
PMP # setenv bootcmd bootm 0x9fd00000
PMP # saveenv
```

然后重启 PMP 板（按 SW2 开关）即可。地址 0xbfd00000 和 0x9fd00000 指向相同的 NOR FLASH 地址，区别在于 0xbfd00000 为 CPU 不可 cache 的地址，而 0x9fd0000 为 CPU 可 cache 的地址。

3.2 配置 CELinux 和设备驱动

我们提供的 CELinux 包含了增强的内核功能和 PMP 的设备驱动程序。本节将为您简单介绍如何配置 CELinux 和设备的驱动程序。

3.2.1 抢占式内核

CELinux 内核采用了抢占式调度功能，极大的提供了系统进程调度的实时性，配置 CELinux 选择 CONFIG_PREEMPT=y 来支持该功能，如下：

```
[General setup]
```

```
    [•]y  [ ]-  [ ]n     Preemptible Kernel
```

3.2.2 动态电源管理

CELinux 支持动态电源管理(Dynamic Power Management)。DPM 提供了 CPU 变频和睡眠/唤醒的支持，以节省系统的功耗。选择以下配置：

```
[Power Management]
    [•]y  [ ]-  [ ]n     Power Management support
    [•]y  [ ]-  [ ]n     Enable Dynamic Power Management Support
    [•]y  [ ]-  [ ]n     Advanced Power Management (APM) Support
```

3.2.3 RTC 驱动

PMP 采用 PHILIPS 的 PCF8563 为 RTC 时钟，配置 CELinux 如下：

```
[Character devices]
    [•]y  [ ]-  [ ]n     Philips PCF8563 Real Time Clock (I2C Bus)
```

3.2.4 MTD 设备驱动

CELinux 分别支持基于 NOR FLASH 和 NAND FLASH 的 MTD 驱动，请参考 PMP 的缺省配置 defconfig-pmp。

3.2.5 LCD Framebuffer

JZ4730 处理器集成了 LCD 控制器，可以直接连接各种 TFT/STN 的 LCD 屏，配置 CELinux 如下：

```
[Console drivers]
  [Frame-buffer support]
    [•]y  [ ]-  [ ]n     Support for frame buffer devices
    [•]y  [ ]-  [ ]n     JzSOC OnChip LCD controller support
    [•]y  [ ]-  [ ]n     SAMSUNG LTP400WQF01 TFT panel (480x272)
```

3.2.6 Audio

JZ4730 处理器集成了 AC97 和 I2S 音频控制器，PMP 支持 AC97 方式，外接了一个 AC97 CODEC，配置 CELinux 如下：

```
[Sound]
    [•]y  [ ]m  [ ]n     Sound card support
    [•]y  [ ]m  [ ]n     JzSOC AC97 sound
```

3.2.7 MMC/SD

JZ4730 处理器集成了 MMC 和 SD 卡控制器，配置 CELinux 如下：

```
[MMC/SD Card support]
    [•]y  [ ]m  [ ]n     MMC support
    [•]y  [ ]m  [ ]n     JZ4730 MMC/SD
```

3.2.8 JFFS2 和 YAFFS 文件系统

配置 CELinux 以支持 JFFS2 和 YAFFS 文件系统，如下：

```
[File systems]
    [•]y  [ ]m  [ ]n  Yet Another Flash Filing System (YAFFS) support
    [•]y  [ ]m  [ ]n  Journaling Flash File System v2 (JFFS2) support
```

3.2.9 USB Host and Gadget

JZ4730 处理器支持 USB 1.1 的 HOST 和 DEVICE 控制器，CELinux 的配置如下：

```
[USB support]
    [•]y  [ ]m  [ ]n  Support for USB
    [•]y  [ ]m  [ ]n  OHCI(Compaq…) support
    [•]y  [ ]m  [ ]n  JzSOC OnChip OHCI-compatible host interface
support
    [Support for USB gadgets]
    [ ]y  [•]m  [ ]n  Support for USB Gadgets
    [JzSOC-UDC]       USB Peripheral Controller Driver
    [ ]y  [•]m  [ ]n  Gadget Zero (DEVELOPMENT)
    [ ]y  [•]m  [ ]n  File-backed Storage gadget (DEVELOPMENT)
```

USB Gadget 驱动采用模块方式，通过 insmod 来加载。

3.2.10 Camera 设备驱动

JZ4730 处理器集成了 Camera 接口，支持 8 位数据线，支持 2048x2048 像数。CELinux 提供了通用的驱动程序接口，包含两个驱动：一个是处理器的 Camera 接口驱动（drivers/char/jzchar/cim.c），另一个是 sensor 的驱动(drivers/char/jzchar/sensor.c)。对 sensor 的配置主要由应用程序来完成，请参考 pmp utility 包的代码。

```
[Character devices]
    [•]y  [ ]m  [ ]n  JzSOC char devices support
    [•]y  [ ]m  [ ]n  JzSOC Camera Interface Module(CIM) support
    [•]y  [ ]m  [ ]n  JzSOC Common CMOS Camera Sensor driver
```

3.2.11 Touchpanel 驱动

```
[Character devices]
   [•]y  [ ]m  [ ]n  JzSOC char devices support
      [•]y  [ ]m  [ ]n  JzSOC touchpanel driver support
      [UCB1400]         Touchpanel codec type
```

3.2.12 PMP 字符设备

PMP 字符设备支持 PMP 的特殊功能，如开机/关机，按键等。

```
[Character devices]
   [•]y  [ ]m  [ ]n  JzSOC char devices support
      [•]y  [ ]m  [ ]n  PMP board poweroff support
      [•]y  [ ]m  [ ]n  PMP board key support
```

4 根文件系统

编译好内核之后，还必须有根文件系统（root filesystem）才能使一个 Linux 系统正常运行。根文件系统用于存放系统运行期间所需的应用程序、脚本、配置文件等。本节将介绍 PMP 根文件系统的功能，以及如何根据自己系统的需求来制作根文件系统。

4.1 PMP 根文件系统

我们为您提供了 PMP 的根文件系统(pmp-root.tar.bz2)。其中包含了 init 进程的启动脚本以及相关工具和应用。下面将介绍这些工具和使用方法。

4.1.1 启动脚本 rcS

启动脚本为 etc/init.d/rcS，主要完成本地文件系统的挂载以及系统的初始化，如启动 telnet server (telnetd)，启动 DPM 后台进程(dpmd)等。

4.1.2 DPM 后台进程

DPM 后台进程 dpmd 用来监测系统的运行状况，并根据不同的状况设定 CPU 的工作频率。

启动 dpmd 的步骤如下：

1） 配置 Linux 内核支持 CONFIG_DPM
2） 启动 DPM 初始化脚本/usr/local/bin/dpm_init
3） 根据系统修改 dpmd 的配置文件/usr/local/etc/dpmd.conf
4） 启动 dpmd

配置文件/usr/local/etc/dpmd.conf 定义了系统运行状态的参数，dpmd 将根据这些参数来设置 CPU 的工作频率。现分别介绍如下：

POLICY_HIGH=XXX

DPM 初始化脚本定义了多个 POLICY，'XXX'必须是其中的一项。POLICY_HIGH 定义了 CPU 最高工作频率对应的 POLICY。

POLICY_LOW=XXX

DPM 初始化脚本定义了多个 POLICY，'XXX'必须是其中的一项。POLICY_LOW 定义了 CPU 最低工作频率对应的 POLICY。

POLL_INTERVAL=N

POLL_INTERVAL 定义了 dpmd 读取和计算系统运行状态的时间间隔，单位是“秒”。如设置“POLL_INTERVAL=2”表示每 2 秒钟计算一次系统运行状态，并根据状态值设定合适的 CPU 工作频率。

CPU_INTERVAL=M-N
SUSPEND_INTERVAL=N
PROGRAMS=XXX,XXX,XXX

CPU_INTERVAL/SUSPEND_INTERVAL/PROGRAMS 分别定义不同的规则，dpmd 将根据这些规则来计算系统的运行状态。

CPU_INTERVAL 定义了 CPU 工作于 POLICY_LOW（最低工作频率）的 CPU 负载值，’M’-‘N’为 CPU 负载的百分比值。如定义“CPU_INTERVAL=0-15”表示当 CPU 负载位于 0%和 15%之间时，CPU 将工作于最低工作频率。当 CPU 负载大于 CPU_INTERVAL 定义的数值时，CPU 将工作于 POLICY_HIGH(最高工作频率)。

SUSPEND_INTERVAL 定义了 CPU 进入睡眠所需的 IDLE 时间间隔，单位为“秒”。当 CPU 负载位于 CPU_INTERVAL 所定义的数值之间时表示 CPU 处于 IDLE 状态。当 IDLE 时间间隔大于 SUSPEND_INTERVAL 定义的数值时，CPU 将进入睡眠状态。

PROGRAMS 定义了 CPU 工作于 POLICY_HIGH（最高工作频率）的进程。可以定义多个进程，不同进程用逗号“,”隔开。如定义“PROGRAMS=mpegplayer,madplay”表示当系统运行进程 mpegplay 或 madplay 时，CPU 将工作于最高工作频率。

我们您提供了 dpmd 的源代码，您可以根据自己的需要修改代码来定义不同的规则。

4.1.3 系统时钟和硬件时钟

PMP 的根文件系统支持对系统时钟和硬件时钟(RTC 时钟)操作，主要命令及使用方法如下：

显示系统时钟：

```
# date
```

显示硬件时钟：

```
# hwclock -r
```

更新系统时钟为硬件时钟：

```
# hwclock -s
```

更新硬件时钟为系统时钟：

```
# hwclock -w
```

修改系统时钟，格式为 # date MMDDhhmmYYYY.ss。如以下命令更新系统时钟为 2006 年 06 月 29 日 12 时 32 分 08 秒，并同步硬件时钟为系统时钟：

```
# date 062912322006.08
# hwclock -w
```

4.1.4 Audio 测试工具

MP3 播放工具：

```
# madplay test.mp3
```

录音和播放录音工具：

```
# vrec –s 48000 –b 16 sample1
# vplay –s 48000 –b 16 sample1
```

4.1.5 Camera 图像采集工具

当配置 Linux 内核支持 CIM 和 Sensor 驱动时，执行以下命令进行测试：

```
# docapture
```

4.1.6 MMC/SD 测试

当配置 Linux 内核支持 MMC/SD 驱动时，连接 MMC 或 SD 卡，Linux 启动后将可以识别和支持对其的操作，比如：

```
# fdisk –l /dev/mmca
# mount –t vfat /dev/mmca1 /mnt
```

4.1.7 USB Gadget 测试

编译 USB Gadget 模块（jz_udc.o 和 g_file_storage.o）并加载：

```
# insmod jz_udc.o
# insmod g_file_storage.o file=/dev/mmca
```

这里指定文件存储设备名为/dev/ram0。连接 PMP 的 USB port0 与主机 USB 端口，此时的 PMP 模拟为 USB mass storage 设备。

4.1.8 QTOPIA/QTE 图形引擎

启动 runqpe 运行 QTOPIA/QTE 图形引擎，如下：

```
# runqpe
```

4.2 BusyBox

嵌入式系统由于存储空间的限制，使得其对程序大小有严格的限制。特别是在制作根文件系统时，更要注意。而使用 BusyBox 就可以大大的简化根文件系统的制作。编译安装后的 BusyBox 只有一个二进制可执行文件 busybox，它实现了几乎所有常用、必须的应用程序（比如 init，shell，getty，ls，cp 等等），而这些应用程序都以符号链接的形式存在。对用户来说，执行命令的方法并没有改变，命令行调用会作为一个参数传给 busybox，即可完成相应的功能。使用 BusyBox 制作的根文件系统既可以节省大量的空间，还节省了大量的交叉编译的工作。

4.2.1 配置和编译 BusyBox

请登陆 BusyBox 的官方网站（http://www.busybox.net）下载起源码包。下面以 busybox-1.1.3 版本为例说明编译和安装 BusyBox 的过程。

编译 BusyBox 的过程类似于编译 Linux 内核的过程：

```
# make defconfig
# make menuconfig
# make
# make install
```

首先，对 BusyBox 进行配置。

```
# make menuconfig
–––––––––––––––––––––––– BusyBox Configuration ––––––––––––––––––––––––
      Busybox Settings  –––>
––– Applets
      Archive Utilities  –––>
      Coreutils  –––>
      Console Utilities  –––>
```

```
    Debian Utilities  ---->
    Editors  ---->
    Finding Utilities  ---->
    Init Utilities  ---->
    Login/Passwd Management Utilities  ---->
    Linux EXT2 FS Progs  ---->
    Linux Module Utilities  ---->
    Linux System Utilities  ---->
    Miscellaneous Utilities  ---->
    Networking Utilities  ---->
    Process Utilities  ---->
    Shells  ---->
    System Logging Utilities  ---->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File

Busybox Settings  ---->
    General Configuration  ---->
    Build Options  ---->
        [ ] Build BusyBox as a static binary (no shared libs)
        [*] Build with Large File Support (for accessing files > 2 GB)
        [*] Do you want to build BusyBox with a Cross Compiler?
        (mipsel-linux-) Cross Compiler prefix
        ( ) Any extra CFLAGS options for the compiler?
        [ ] Compile all sources at once
```

这里有几个选项要注意：动态编译，使用共享库；设置交叉编译器的路径和前缀，根据我们的情况就要设置为 mipsel-linux-。

```
    Install Options  ---->
        [*] Don't use /usr
        (./_install) BusyBox installation prefix
```

这里设置安装路径(./_install)。不使用/usr 的意思是指全部安装在/bin 和/sbin，不在/usr/bin 和/usr/sbin 下面安装程序。

```
    Linux Module Utilities  ---->
        [*] insmod
        [ ] In kernel memory optimization (uClinux only)
        [*] rmmod
        [*] lsmod
        [*] modprobe
        [*] Support version 2.2.x to 2.4.x Linux kernels
```

…

注意：配置 module 工具时不选择[In kernel memory optimization]，在编译模块.o 时不使用 GCC 的编译选项”-g”。

```
Shells  ---->
    Choose your default shell (ash)  ---->
```

这里选择 ash 为缺省的 shell。

```
Networking Utilities  ---->
    [ ] Enable IPv6 support
    [*] telnetd
    [ ]       Support call from inet only
```

这里的设置使得 BusyBox 可以支持 telnet server，这样可以通过远程 telnet 到目标机。

其余部分用来配置所实现的工具，可以根据需要选择，或者使用缺省配置即可。

然后就可以编译和安装了。

4.2.2 根文件系统内容

按照上面步骤安装的根文件系统（/nfsroot/mips/pmproot）有下面几个目录和文件：bin，sbin，lib，和 linuxrc。接下来还需要添加一些目录和内容，才能使 BusyBox 工作。关键是 dev，etc 和必要的 lib 共享库。

```
# cd /nfsroot/mips/pmproot
# mkdir dev etc lib mnt opt proc root tmp usr var
# ls
bin dev etc lib linuxrc mnt opt proc root sbin tmp usr var
```

bin 和 sbin 下面就是 BusyBox 编译安装的内容，dev 和 etc 可以直接复制 pmp-root.tar.bz2 下的内容。lib 目录存放动态连接库.so 文件，如 C 库，Math 库，C++库等。可以参考 pmp-root.tar.bz2 下的内容有选择的复制。其他目录则可以用来存放用户自己开发的应用程序。

当根文件系统制作完成后，还需要一个把其放到目标板的 NOR 或 NAND FLASH 上的过程。一种方式是制作成 initrd，把 initrd 和 Linux 内核编译在一起；另外一种方式是在目标板的 NOR 或 NAND FLASH 上构建文件系统（如 JFFS2），然后把根文件系统的内容复制到 FLASH 分区上。

制作 initrd 时请参考下面步骤：

```
# dd if=/dev/zero of=initrd bs=8192k count=1
```

```
# mke2fs -F -m0 -b 1024 initrd
# mount -t ext2 -o loop initrd /mnt
# cp -a /nfsroot/mips/pmproot/* /mnt/
# umount /mnt
# gzip -9 initrd
```

把 initrd.gz 拷贝到 CELinux 目录下，并取名为 ramdisk.gz：

```
# cp initrd.gz /CELinux/arch/mips/ramdisk/ramdisk.gz
```

然后配置 CELinux 如下：

```
[Block devices]
    [•]y  [ ]m  [ ]n  RAM disk support
    [8192]            Default RAM disk size
    [•]y  [ ]m  [ ]n  initial RAM disk (initrd) support
    [•]y  [ ]m  [ ]n  Embed root filesystem ramdisk into the kernel
    [ramdisk.gz]      Filename of gziped ramdisk image
```

重新编译 Linux。然后在 U-Boot 中设置 Linux 的命令行参数为“root=/dev/ram0”。

制作 JFFS2 文件系统时请参考下面的步骤：

在主机端创建 JFFS2 文件系统的映像：

```
# mkfs.jffs2 -p –e 0x1000 –d /nfsroot/mips/pmproot –o pmproot.jffs2
```

其中：

-d /nfsroot/mips/pmproot：根文件系统的内容放在/nfsroot/mips/pmproot 目录中。
-e 0x10000：FLASH 的 eraseblock 大小为 0x10000 字节。
-p：以 0xff 填充文件尾部的空间对齐 eraseblock。
-o pmproot.jffs2：输出映像文件名。

将映像文件复制到 NFS ROOT 的根目录（这里是/nfsroot/pmp-root/），然后配置 U-Boot 使用 /nfsroot/pmp-root 作为 NFS 文件系统，启动 Linux 后，在 Target 端把 JFFS2 文件系统的映像复制到 MTD 分区上：

```
# cat /proc/mtd
# flash_eraseall –j /dev/mtd0
# cp pmproot.jffs2 /dev/mtd0
```

然后在 U-Boot 中设置 Linux 的命令行参数为“root=/dev/mtdblock0 rw”，并重新启动 Linux。

另外一种方法是采用 NFS ROOT 启动目标板，在目标板上利用 JFFS2 工具来制作文件系统，如下：

```
# flash_eraseall –j /dev/mtd0
# mount –t jffs2 /dev/mtdblock0 /mnt
```

然后，复制所需要的内容到/mnt 下：

```
# cp –a bin dev etc linux sbin proc … /mnt
# umount
```

flash_eraseall 使用-j 选项来格式化/dev/mtd0 为 JFFS2 文件系统，只需要格式化一次，之后就可以直接使用 mount 命令挂载了。