一、移植前的编译测试

1.首先从网上下载原生内核源码 linux-2.6.32，修改主 Makefile 文件：将 ARCH 修改为 ARM，CROSS_COMPILE 修改为 arm-linux-

```
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH            ?= arm
CROSS_COMPILE   ?= arm-linux-
```

2.按照 s3c2410 的默认配置对内核进行配置：make s3c2410_defconfig

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# ls
arch    COPYING  crypto         drivers   fs        init  Kbuild  lib          Makefile  net
block   CREDITS  Documentation  firmware  include   ipc   kernel  MAINTAINERS  mm        README
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make s3c2410_defconfig
```

3.进行编译：**make**

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# ls
arch    COPYING  crypto         drivers   fs        init  Kbuild  lib          Makefile  net
block   CREDITS  Documentation  firmware  include   ipc   kernel  MAINTAINERS  mm        README
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make s3c2410_defconfig
#
# configuration written to .config
#
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make
```

4.发现编译过程出现 timeconst.pl 的 373 行编译错误：

```
  CC      kernel/fork.o
  CC      kernel/exec_domain.o
  CC      kernel/panic.o
  CC      kernel/printk.o
  CC      kernel/cpu.o
  CC      kernel/exit.o
  CC      kernel/itimer.o
  TIMEC   kernel/timeconst.h
Can't use 'defined(@array)' (Maybe you should just omit the defined()?) at kernel/timeconst.pl line 373.
/media/luowei/学习/just-for-fun/linux-2.6.32/kernel/Makefile:129: recipe for target 'kernel/timeconst.h' failed
make[1]: *** [kernel/timeconst.h] Error 255
Makefile:878: recipe for target 'kernel' failed
make: *** [kernel] Error 2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32#
```

对相应文件进行修改：**vim kernel/timeconst.pl**

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# ls
arch      CREDITS        drivers   include  Kbuild  MAINTAINERS  modules.order  REPORTING-BUGS
block     crypto         firmware  init     kernel  Makefile     net            samples
COPYING   Documentation  fs        ipc      lib     mm           README         scripts
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# vim kernel/timeconst.pl
```

末尾行模式输入：**set nu** 显示行号

```
 36                 31250,1,
 37                 '0x8637bd06','0
 38                 1,31250,
 39         ], 48 => [
 40                 '0xa6aaaaab','0
 41                 125,6,
 42                 '0xc49ba5e4','0
:set nu
```

末尾行模式直接输入行号后光标直接跳到所在行：

```
372         @val = @{$canned_values{$hz}};
373         if (!defined(@val)) {
374                 @val = compute_values($hz);
375         }
376         output($hz, @val);
377 }
378 exit 0;
:373
```

将 373 的 defined 去掉后保存退出。

```
371
372         @val = @{$canned_values{$hz}};
373         #if (!defined(@val)) {  #error code
374         if (!(@val)) {
375                 @val = compute_values($hz);
376         }
377         output($hz, @val);
378 }
379 exit 0;
:wq
```

5.可能出现如下错误：

```
  CC [M]  net/ipv4/netfilter/ipt_ah.o
  CC [M]  net/ipv4/netfilter/ipt_ecn.o
  CC [M]  net/ipv4/netfilter/ipt_CLUSTERIP.o
make[3]: *** No rule to make target 'net/ipv4/netfilter/ipt_ECN.c', needed by 'net/ipv4/netfilter/ipt_ECN.o'。 停止。
scripts/Makefile.build:365: recipe for target 'net/ipv4/netfilter' failed
make[2]: *** [net/ipv4/netfilter] Error 2
scripts/Makefile.build:365: recipe for target 'net/ipv4' failed
make[1]: *** [net/ipv4] Error 2
Makefile:878: recipe for target 'net' failed
make: *** [net] Error 2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32#
```

是因为内核是在 windows 环境解压产生文件覆盖造成的：

刚刚编译过的linux内核，去windows转了一圈改进了小块地方，再上ubuntu就不能再编译了
纠结了很久，工程也一次一次重建了，还是不ok。悲催阿...

仔细查看出错原因，+BD,终于ok了，下面是出错原因和解决办法：

问题：

```
make[3]: *** No rule to make target 'net/ipv4/netfilter/ipt_ecn.c', needed by
'net/ipv4/netfilter/ipt_ecn.o'
make[2]: *** [net/ipv4/netfilter] Error 2
make[1]: *** [net/ipv4] Error 2
make: *** [net] Error 2
```

解决：之前以为是权限问题，一再给根目录777权限，还是出错。
后经多方搜索发现是由于windows与linux实现机制原因
在windows下，文件不分大小写，结果linux下名称相同，但有大小写区别的文件，在windows成一个文件，其他被覆盖。悲剧阿....
针对上面错误，仔细查证发现，果然ipt_ecn.c被同路径下ipt_ECN.c覆盖。
建议：如果要在windows下改源码，编译时尽量只去覆盖已经修改的源码，linux源码下有很多会被windows认为是同名文件的文件。切勿为了逃避单文件覆盖的繁琐，而全部贴过来再编译，这样子只会更麻烦。
感悟：以前觉得windows真好用，自从它莫名其妙覆盖了同名（大小写敏感）文件后，只想说句：windows真坑人

请一定要在 linux 环境下解压源码包：**tar xjf linux-2.6.32.tar.bz2**

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# cd ..
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# ls
linux-2.6.32  linux-2.6.32.tar.bz2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# rm -r linux-2.6.32
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# ls
linux-2.6.32.tar.bz2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# tar xjf linux-2.6.32.tar.bz2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# ls
linux-2.6.32  linux-2.6.32.tar.bz2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun#
```

6.如果没有其他错误，编译成功后会在/arch/arm/boot/目录下产生编译的镜像文件：

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# cd arch/arm/boot/
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/boot# ls
bootp  compressed  Image  install.sh  Makefile  zImage
```

二、向 mini2440 平台的移植

1.查看 mini2440 平台的机器码：**cat arch/arm/tools/mach-types**

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/tools# ls
gen-mach-types  mach-types  Makefile
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/tools# cat mach-types
# Database of machine macros and numbers
#
# This file is linux/arch/arm/tools/mach-types
#
# Up to date versions of this file can be obtained from:
#
#   http://www.arm.linux.org.uk/developer/machines/download.php
#
# Please do not send patches to this file; it is automatically generated!
# To add an entry into this database, please see Documentation/arm/README,
# or visit:
#
#   http://www.arm.linux.org.uk/developer/machines/?action=new
#
# Last update: Wed Nov 25 22:14:58 2009
#
# machine_is_xxx        CONFIG_xxxx             MACH_TYPE_xxx           number
#
ebsa110                 ARCH_EBSA110            EBSA110                 0
riscpc                  ARCH_RPC                RISCPC                  1
nexuspci                ARCH_NEXUSPCI           NEXUSPCI                3
ebsa285                 ARCH_EBSA285            EBSA285                 4
netwinder               ARCH_NETWINDER          NETWINDER               5
cats                    ARCH_CATS               CATS                    6
tbox                    ARCH_TBOX               TBOX                    7
```

如下图 mini2440 已经被申请好了机器码 1999，通过 bootloader 传入的机器码 MACH_TYEP 确定启动哪种目标平台，在 u-boot/include/asm-arm/mach-types.h 中也可以看到机器码的宏定义
#define MACH_TYPE_MINI2440 1999

```
f5d8231_4_v2            MACH_F5D8231_4_V2       F5D8231_4_V2            1996
q2440                   MACH_Q2440              Q2440                   1997
qq2440                  MACH_QQ2440             QQ2440                  1998
mini2440                MACH_MINI2440           MINI2440                1999
colibri300              MACH_COLIBRI300         COLIBRI300              2000
jades                   MACH_JADES              JADES                   2001
spark                   MACH_SPARK              SPARK                   2002
```

2.复制 mach-smdk2440.c 文件为 mach-mini2440 文件：
首先要删除已有的文件：**cd arch/arm/mach-s3c2440**
rm mach-mini2440.c
然后复制一份:**cp mach-smdk2440.c mach-mini2440.c**

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# ls
built-in.o  dma.c  dsc.o  Kconfig       mach-at2440evb.c  mach-mini2440.o  mach-osiris.c  mach-rx3715.o    Makefile       s3c2440.o
clock.c     dma.o  irq.c  mach-anubis.c  mach-at2440evb.o  mach-nexcoder.c  mach-osiris.o  mach-smdk2440.c  modules.order
clock.o     dsc.c  irq.o  mach-anubis.o  mach-mini2440.c   mach-nexcoder.o  mach-rx3715.c  mach-smdk2440.o  s3c2440.c
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# rm mach-mini2440.c
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# ls
built-in.o  dma.c  dsc.o  Kconfig       mach-at2440evb.c  mach-nexcoder.c  mach-osiris.o  mach-smdk2440.c  modules.order
clock.c     dma.o  irq.c  mach-anubis.c  mach-at2440evb.o  mach-nexcoder.o  mach-rx3715.c  mach-smdk2440.o  s3c2440.c
clock.o     dsc.c  irq.o  mach-anubis.o  mach-mini2440.o   mach-osiris.c    mach-rx3715.o  Makefile         s3c2440.o
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# cp mach-smdk2440.c mach-mini2440.c
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# ls
built-in.o  dma.c  dsc.o  Kconfig       mach-at2440evb.c  mach-mini2440.o  mach-osiris.c  mach-rx3715.o    Makefile       s3c2440.o
clock.c     dma.o  irq.c  mach-anubis.c  mach-at2440evb.o  mach-nexcoder.c  mach-osiris.o  mach-smdk2440.c  modules.order
clock.o     dsc.c  irq.o  mach-anubis.o  mach-mini2440.c   mach-nexcoder.o  mach-rx3715.c  mach-smdk2440.o  s3c2440.c
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32/arch/arm/mach-s3c2440# vim mach-mini2440.c
```

3.修改 mach-mini2440.c 文件：

（1）改晶振频率 12M：**s3c24xx_init_clocks(12000000);**

（2）注释掉初始化函数：**//smdk_machine_init();**

（3）修改开发板信息：**MACHINE_START(MINI2440,“自定义信息”);**

（4）把全局所有字符串中的 smdk2440 替换为 mini2440：**%s/smdk2440/mini2440/g**

```
160 static void __init smdk2440_map_io(void)
161 {
162         s3c24xx_init_io(smdk2440_iodesc, ARRAY_SIZE(smdk2440_iodesc));
163         //s3c24xx_init_clocks(16934400);修改为下一行
164         s3c24xx_init_clocks(12000000);//晶振频率12M
165         s3c24xx_init_uarts(smdk2440_uartcfgs, ARRAY_SIZE(smdk2440_uartcfgs));
166 }
167
168 static void __init smdk2440_machine_init(void)
169 {
170         s3c24xx_fb_set_platdata(&smdk2440_fb_info);
171         s3c_i2c0_set_platdata(NULL);
172
173         platform_add_devices(smdk2440_devices, ARRAY_SIZE(smdk2440_devices));
174         //smdk_machine_init();初始化函数注释掉，以后会自己编写
175 }
176
177 //MACHINE_START(S3C2440, "SMDK2440")修改为下一行
178 MACHINE_START(MINI2440, "luowei's mini2440 development board")//cat proc/cpuinfo可以看到此信息
179         /* Maintainer: Ben Dooks <ben@fluff.org> */
180         .phys_io        = S3C2410_PA_UART,
181         .io_pg_offst    = (((u32)S3C24XX_VA_UART) >> 18) & 0xfffc,
182         .boot_params    = S3C2410_SDRAM_PA + 0x100,
183
:%s/smdk2440/mini2440/g
```

3.编译测试，使用 linux 官方自带的 mini2440 配置，进行编译产生 zImage：

make mini2440_defconfig

make zImage

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make mini2440_defconfig
#
# configuration written to .config
#
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make zImage
```

可以下载到板子中进行测试，内核可以正常启动，但是没有大部分硬件驱动和文件系统，因此目前还无法登录。

4.通过 make menuconfig 查看所要启动的设备型号：

（menuconfig 功能需要提前安装：apt -get install libncurses5-dev）

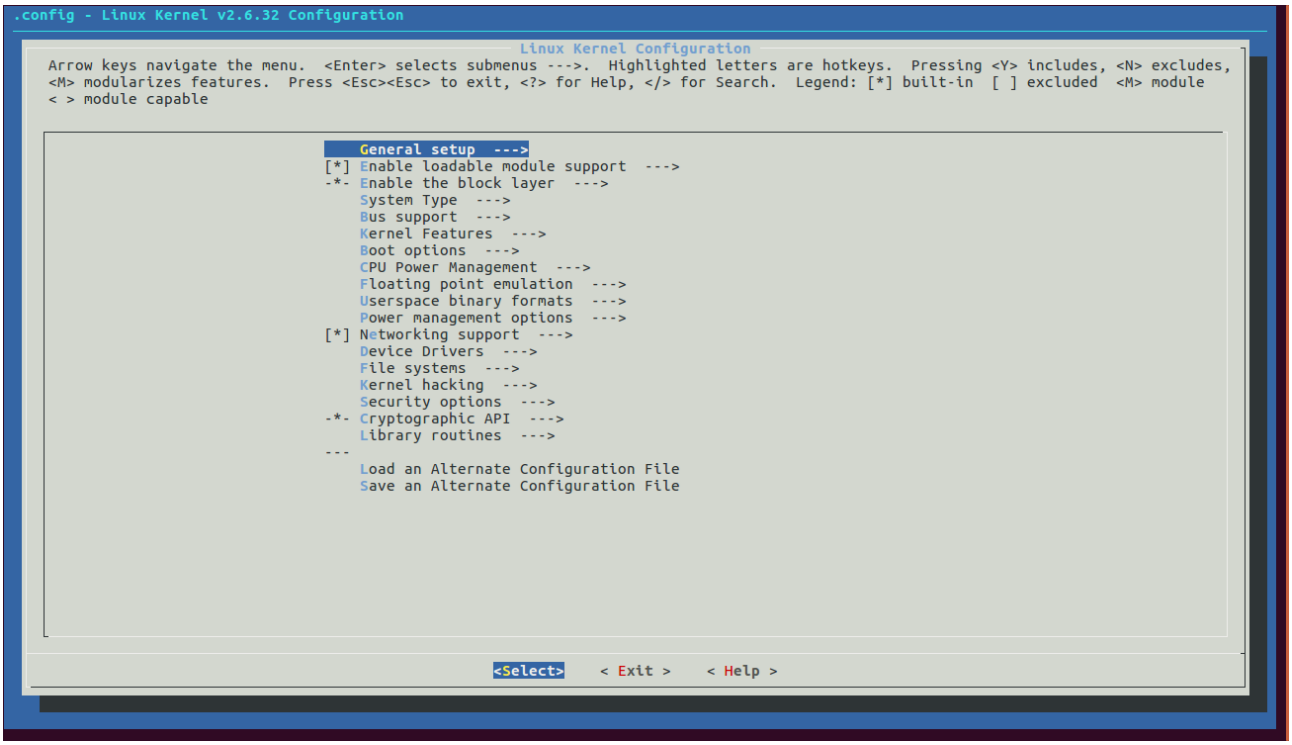
```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32# make menuconfig
```

稍等片刻会进入文字菜单配置界面：

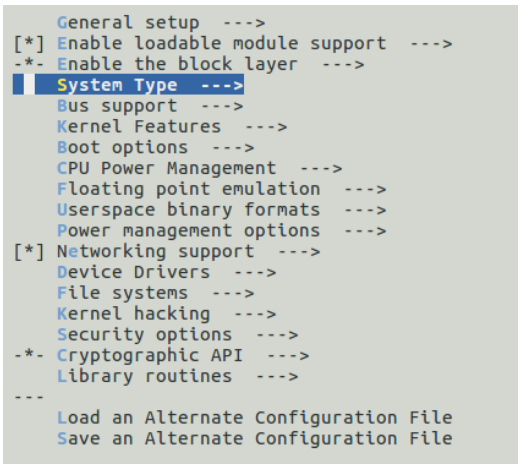
[]和[*]代表两种选，通过空格切换

<>、<*>和<M>代表三种选择，<M>代表模块

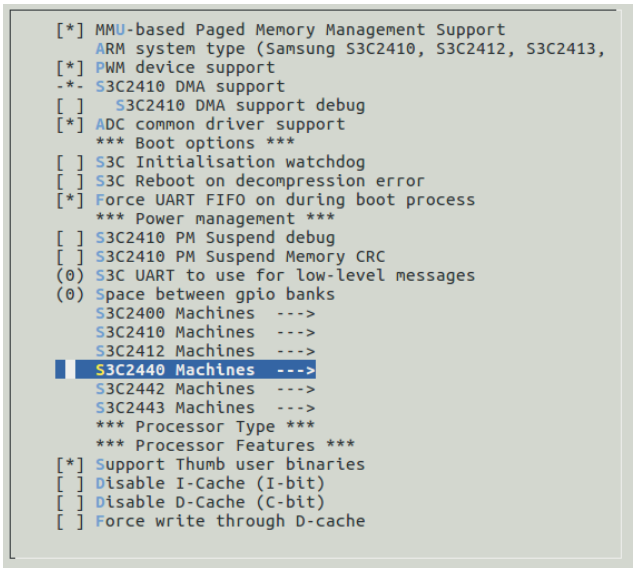
--→代表还有子菜单，---或-*-



按上下键选择 System Type，按回车键进入：



选择 S3C2440 Machines 进入：



发现已经勾选了设备 MINI2440 development board

```
[ ] Simtec Electronics ANUBIS
[ ] Simtec IM2440D20 (OSIRIS) module
[ ] HP iPAQ rx3715
[ ] SMDK2440
[ ] NexVision NEXCODER 2440 Light Board
[ ] Avantech AT2440EVB development board
[*] MINI2440 development board
```

此菜单从何而来，首先看 arch/arm/mach-s3c2440 目录下的 Kconfig 文件：

```
linux-2.6.32# vim arch/arm/mach-s3c2440/Kconfig
```

在此处可以修改菜单中选项的文本信息：

```
config MACH_MINI2440
        #bool "MINI2440 development board"此处菜单显示信息可以自定义
        bool "LuoWei mini2440 development board"
        select CPU_S3C2440
        select EEPROM_AT24
        select LEDS_TRIGGER_BACKLIGHT
        select SND_S3C24XX_SOC_S3C24XX_UDA134X
        select S3C_DEV_NAND
        select S3C_DEV_USB_HOST
        help
          Say Y here to select support for the MINI2440. Is a 10cm x 10cm board
          available via various sources. It can come with a 3.5" or 7" touch LCD.

endmenu
```

然后看此文件目录下的 Makefile 文件：

```
linux-2.6.32# vim arch/arm/mach-s3c2440/Makefile
```

Makefile 中的配置定义 CONFIG_MACH_MINI2440 将配置文件.config 和实际代码 mach-mini2440.c 联系起来。

```
# Machine support

obj-$(CONFIG_MACH_ANUBIS)       += mach-anubis.o
obj-$(CONFIG_MACH_OSIRIS)       += mach-osiris.o
obj-$(CONFIG_MACH_RX3715)       += mach-rx3715.o
obj-$(CONFIG_ARCH_S3C2440)      += mach-smdk2440.o
obj-$(CONFIG_MACH_NEXCODER_2440) += mach-nexcoder.o
obj-$(CONFIG_MACH_AT2440EVB) += mach-at2440evb.o
obj-$(CONFIG_MACH_MINI2440) += mach-mini2440.o
```

可以看到源码根目录.config 文件中定义了 CONFIG_MACH_MINI2440=y，代表默认勾选，会编译在内核镜像中：（obj-m 代表模块，不会编译进内核镜像，但是会生成.ko 文件在系统运行过程中装载模块，没有在.config 文件中定义的配置项则不会被编译，一般会写一条注释# CONFIG_XXX is not set）

```
#
# S3C2440 Machines
#
# CONFIG_MACH_ANUBIS is not set
# CONFIG_MACH_OSIRIS is not set
# CONFIG_MACH_RX3715 is not set
# CONFIG_ARCH_S3C2440 is not set
# CONFIG_MACH_NEXCODER_2440 is not set
# CONFIG_MACH_AT2440EVB is not set
CONFIG_MACH_MINI2440=y

#
# S3C2442 Machines
#
# CONFIG_MACH_NEO1973_GTA02 is not set

#
# S3C2443 Machines
#
# CONFIG_MACH_SMDK2443 is not set
```

三、移植 yaffs2 文件系统

1.修改 mach-s3c2440 添加 nand flash 的相关信息：

```
linux-2.6.32$ vim arch/arm/mach-s3c2440/mach-mini2440.c
```

（1）添加头文件

```
//添加以下头文件
#include <linux/mtd/mtd.h>//1
#include <linux/mtd/nand.h>//1
#include <linux/mtd/nand_ecc.h>//1
#include <linux/mtd/partitions.h>//1
#include <plat/nand.h>//1
```

（2）添加 nand flash 分区信息，nand flash 设置表，nand flash 本身属性

这三个信息，第 2 个包含了第 1 个，第 3 个包含了第 2 个，因此只需要在初始化函数中实际配置第 3 条信息即可。

```
//////////////////////添加如下信息
static struct mtd_partition mini2440_default_nand_part[] = {
[0] = { .name = "supervivi", //bootloader所在的分区,可以放置 u-boot, supervivi 等内容,对应/dev/mtdblock0
        .offset = 0,//这里的单位都是Byte
        .size = 0x00040000,},//256KB
[1] = { .name= "param", //supervivi 的参数区,也属于bootloader的一部分,如果u-boot比较大,可以把此区域覆盖掉,不会影响系统启动,对应/dev/mtdblock1
        .offset = 0x00040000,
        .size = 0x00020000,},//128KB
[2] = { .name= "Kernel", //内核所在的分区,大小为5M,足够放下大部分自己定制的巨型内核了,比如内核使用了更大的Linux Logo 图片等,对应/dev/mtdblock2
        .offset = 0x00060000,
        .size = 0x00500000,},//5MB
[3] = { .name = "root", //文件系统分区,友善之臂主要用来存放 yaffs2 文件系统内容,对应/dev/mtdblock3
        .offset = 0x00560000,
        .size = 1024 * 1024 * 1024,},//1GB
[4] = { .name = "nand", //此区域代表了整片的nand flash,主要是预留使用,比如以后可以通过应用程序访问读取/dev/mtdblock4就能实现备份整片nand flash
        .offset = 0x00000000,
        .size= 1024 * 1024 * 1024,}//1GB
};

//开发板的nand flash设置表,因为板子上只有一片,因此也就只有一个表
static struct s3c2410_nand_set mini2440_nand_sets[] = {
[0] = { .name = "NAND",
        .nr_chips = 1,
        .nr_partitions = ARRAY_SIZE(mini2440_default_nand_part),
        .partitions = mini2440_default_nand_part,}//分区配置
};

//nand flash本身的一些特性,一般需要对照datasheet填写,大部分情况下按照以下参数填写即可
static struct s3c2410_platform_nand mini2440_nand_info = {
        .tacls = 20,
        .twrph0 = 60,
        .twrph1 = 20,
        .nr_sets = ARRAY_SIZE(mini2440_nand_sets),
        .sets = mini2440_nand_sets,
        .ignore_unset_ecc = 1,
};
//////////////////////
```

（3）把 nand flash 设备添加到开发板的设备列表结构中

```
static struct platform_device *mini2440_devices[] __initdata = {
        &s3c_device_usb,
        &s3c_device_lcd,
        &s3c_device_wdt,
        &s3c_device_i2c0,
        &s3c_device_iis,
        &s3c_device_nand,//(添加此行)把nand flash设备添加到开发板的设备列表结构
};
```

（4）在 mini2440_machine_init()函数中添加：

s3c_device_nand.dev.platform_data=&mini2440_nand_info;

```
static void __init mini2440_machine_init(void)
{
        s3c24xx_fb_set_platdata(&mini2440_fb_info);
        s3c_i2c0_set_platdata(NULL);

        platform_add_devices(mini2440_devices, ARRAY_SIZE(mini2440_devices));
        //添加下面这句，否则会在开机时出现错误，内核对nand flash的特性错误地去采用了默认配置
        s3c_device_nand.dev.platform_data=&mini2440_nand_info;
        //smdk_machine_init();初始化函数注释掉，以后会自己编写
}
```

否则在内核启动过程中会出现如下错误：Tacls=4, 39ns Twrph0=8 79ns, Twrph1=8 79ns
发现这个 nand flash 属性配置和我们的配置不符，说明采用了默认配置，我们的配置没有生效，需要通过以上方法在初始化函数中进行配置后生效。

```
fb0: s3c2410fb frame buffer device
s3c2440-uart.0: s3c2410_serial0 at MMIO 0x50000000 (irq = 70) is a S3C2440
s3c2440-uart.1: s3c2410_serial1 at MMIO 0x50004000 (irq = 73) is a S3C2440
s3c2440-uart.2: s3c2410_serial2 at MMIO 0x50008000 (irq = 76) is a S3C2440
brd: module loaded
S3C24XX NAND Driver, (c) 2004 Simtec Electronics
s3c24xx-nand s3c2440-nand: Tacls=4, 39ns Twrph0=8 79ns, Twrph1=8 79ns
Unable to handle kernel NULL pointer dereference at virtual address 00000018
pgd = c0004000
[00000018] *pgd=00000000
Internal error: Oops: 5 [#1]
last sysfs file:
Modules linked in:
CPU: 0    Not tainted  (2.6.32 #0)
PC is at s3c24xx_nand_probe+0x2ec/0x52c
LR is at s3c24xx_nand_probe+0x1b8/0x52c
pc : [<c01ef420>]    lr : [<c01ef2ec>]    psr: 60000013
sp : c3823ef0  ip : 00000000  fp : c01ef11c
r10: c01ee958  r9 : c01ef128  r8 : 00000000
r7 : 00000001  r6 : 00000000  r5 : c388f800  r4 : c39b6900
r3 : 00000001  r2 : c4c00010  r1 : c4c00004  r0 : c388fab8
Flags: nZCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment kernel
Control: c000717f  Table: 30004000  DAC: 00000017
```

至此，mach-mini2440.c 添加 nand flahs 信息的修改工作完成。

2.下载 yaffs2 安装包：git clone git://www.aleph1.co.uk/yaffs2

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# git clone git://www.aleph1.co.uk/yaffs2
正克隆到 'yaffs2'...
remote: Counting objects: 8052, done.
remote: Compressing objects: 100% (5133/5133), done.
remote: Total 8052 (delta 6386), reused 3635 (delta 2834)
接收对象中: 100% (8052/8052), 3.67 MiB | 68.00 KiB/s, 完成.
处理 delta 中: 100% (6386/6386), 完成.
检查连接... 完成。
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# ls
linux-2.6.32  linux-2.6.32.tar.bz2  yaffs2
root@luowei-thinkpad:/media/luowei/学习/just-for-fun# cd yaffs2/
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2# ls
devnotes          patch-ker.sh        yaffs_bitmap.h       yaffs_guts.h          yaffs_packedtags1.c   yaffs_tagsmarshall.h  yaffs_yaffs2.h
Kconfig_multi     README-linux        yaffs_checkptrw.c    yaffs_linux.h         yaffs_packedtags1.h   yaffs_trace.h         yportenv.h
Kconfig_single    README-linux-patch  yaffs_checkptrw.h    yaffs_mtdif.h         yaffs_packedtags2.c   yaffs_verify.c        yportenv_multi.h
linux-tests       utils               yaffs_ecc.c          yaffs_mtdif_multi.c   yaffs_packedtags2.h   yaffs_verify.h        yportenv_single.h
Makefile          yaffs_allocator.c   yaffs_ecc.h          yaffs_mtdif_single.c  yaffs_summary.c       yaffs_vfs_multi.c
Makefile.kernel   yaffs_allocator.h   yaffs_endian.c       yaffs_nameval.c       yaffs_summary.h       yaffs_vfs_single.c
moduleconfig.h    yaffs_attribs.c     yaffs_endian.h       yaffs_nameval.h       yaffs_tagscompat.c    yaffs_yaffs1.c
mtdemul           yaffs_attribs.h     yaffs_getblockinfo.h yaffs_nand.c          yaffs_tagscompat.h    yaffs_yaffs1.h
patches           yaffs_bitmap.c      yaffs_guts.c         yaffs_nand.h          yaffs_tagsmarshall.c  yaffs_yaffs2.c
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2#
```

输入：**./patch-ker.sh c ../linux-2.6.32**，会出现提示信息，需要增加参数 m：
(和友善之臂参考文档不一致，文档中只有参数 c 没有问题，目前应该下载了更新版的 yaffs2)

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2# ./patch-ke
r.sh c ../linux-2.6.32
usage:   ./patch-ker.sh  c/l m/s kernelpath
 if c/l is c, then copy. If l then link
 if m/s is m, then use multi version code. If s then use single version
 code
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2#
```

重新输入：**./patch-ker.sh c m ../linux-2.6.32**，出现如下信息，说明补丁运行成功。

```
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2# ./patch-ker.sh c m ../linux-2.6.32
Updating ../linux-2.6.32/fs/Kconfig
Updating ../linux-2.6.32/fs/Makefile
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/yaffs2#
```

补丁做了 3 件工作：

（1）源码目录 fs 文件夹中多了一个 yaffs 文件：

```
linux-2.6.32/fs$ ls
ext2            generic_acl.o   libfs.c         nilfs2          readdir.c       super.o
ext3            gfs2            libfs.o         nls             readdir.o       sync.c
ext4            hfs             lockd           no-block.c      read_write.c    sync.o
fat             hfsplus         locks.c         notify          read_write.h    sysfs
fcntl.c         hostfs          locks.o         ntfs            read_write.o    sysv
fcntl.o         hpfs            Makefile        ocfs2           reiserfs        timerfd.c
fifo.c          hppfs           mbcache.c       omfs            romfs           timerfd.o
fifo.o          hugetlbfs       mbcache.o       open.c          select.c        ubifs
file.c          inode.c         minix           open.o          select.o        udf
file.o          inode.o         modules.order   openpromfs      seq_file.c      ufs
filesystems.c   internal.h      mpage.c         partitions      seq_file.o      utimes.c
filesystems.o   ioctl.c         mpage.o         pipe.c          signalfd.c      utimes.o
file_table.c    ioctl.o         namei.c         pipe.o          signalfd.o      xattr_acl.c
file_table.o    ioprio.c        namei.o         pnode.c         smbfs           xattr_acl.o
freevxfs        ioprio.o        namespace.c     pnode.h         splice.c        xattr.c
fscache         isofs           namespace.o     pnode.o         splice.o        xattr.o
fs_struct.c     jbd             ncpfs           posix_acl.c     squashfs        xfs
fs_struct.o     jbd2            nfs             posix_acl.o     stack.c         yaffs2
fs-writeback.c  jffs2           nfs_common      proc            stack.o
fs-writeback.o  jfs             nfsctl.c        qnx4            stat.c
fuse            Kconfig         nfsctl.o        quota           stat.o
generic_acl.c   Kconfig.binfmt  nfsd            ramfs           super.c
```

（2）此目录下的 Kconfig 文件，多了一行 source “fs/yaffs2/Kconfig”

```
linux-2.6.32/fs$ vim Kconfig
```

```
source "fs/bfs/Kconfig"
source "fs/efs/Kconfig"
#运行yaffs2补丁自动会增加下面这行
source "fs/yaffs2/Kconfig"
source "fs/jffs2/Kconfig"
# UBIFS File system configuration
source "fs/ubifs/Kconfig"
source "fs/cramfs/Kconfig"
source "fs/squashfs/Kconfig"
source "fs/freevxfs/Kconfig"
```

（3）此目录下的 Makefile，多了一行 obj-$(CONFIG_YAFFS_FS) +=yaffs2/

```
linux-2.6.32/fs$ vim Makefile
```

```
123 obj-$(CONFIG_OCFS2_FS)              += ocfs2/
124 obj-$(CONFIG_BTRFS_FS)              += btrfs/
125 obj-$(CONFIG_GFS2_FS)               += gfs2/
126 obj-$(CONFIG_EXOFS_FS)              += exofs/
127 #运行yaffs2补丁会自动增加下面这行
128 obj-$(CONFIG_YAFFS_FS)              += yaffs2/
```

可能出现的 yaffs2 版本问题：

由于下载的 yaffs2 比文档中版本新的问题，在后来的内核启动过程中发生了如下错误：

Unable to handle kernel paging request at virtual address 72630012

```
yaffs: dev is 32505859 name is "mtdblock3" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs filesystem) on device 31:3.
Freeing init memory: 128K
Unable to handle kernel paging request at virtual address 72630012
pgd = c0004000
[72630012] *pgd=00000000
Internal error: Oops: 3 [#1]
last sysfs file:
Modules linked in:
CPU: 0    Not tainted  (2.6.32 #0)
PC is at yaffs_getxattr+0x2c/0x84
LR is at get_vfs_caps_from_disk+0x50/0xec
pc : [<c0144a00>]    lr : [<c0162428>]    psr: 60000013
sp : c3823ec8  ip : c01449d4  fp : c03de0ac
r10: c3823f80  r9 : 00000002  r8 : c340cc00
r7 : 00000014  r6 : 7263000a  r5 : c3823ee4  r4 : c3823f0c
r3 : 00000014  r2 : f0000010  r1 : c0387007  r0 : c340cc00
Flags: nZCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment kernel
Control: c000717f  Table: 30004000  DAC: 00000017
Process swapper (pid: 1, stack limit = 0xc3822270)
Stack: (0xc3823ec8 to 0xc3824000)
```

解决方法：将补丁自动拷贝的 yaffs2 文件夹删除，采用已经移植好的内核源码目录 fs/下的 yaffs2 文件夹。（查看这新旧版本的 yaffs2 文件夹，发现其中.c 和.h 文件个数和文件名都不一样，造成了兼容性问题）

3.将 yaffs2 文件系统选项勾选

（1）输入：**make menuconfig**，进入 File systems：

```
linux-2.6.32# make menuconfig
```

```
    General setup  --->
[*] Enable loadable module support  --->
-*- Enable the block layer  --->
    System Type  --->
    Bus support  --->
    Kernel Features  --->
    Boot options  --->
    CPU Power Management  --->
    Floating point emulation  --->
    Userspace binary formats  --->
    Power management options  --->
[*] Networking support  --->
    Device Drivers  --->
    File systems  --->
    Kernel hacking  --->
    Security options  --->
-*- Cryptographic API  --->
    Library routines  --->
---
    Load an Alternate Configuration File
    Save an Alternate Configuration File
```

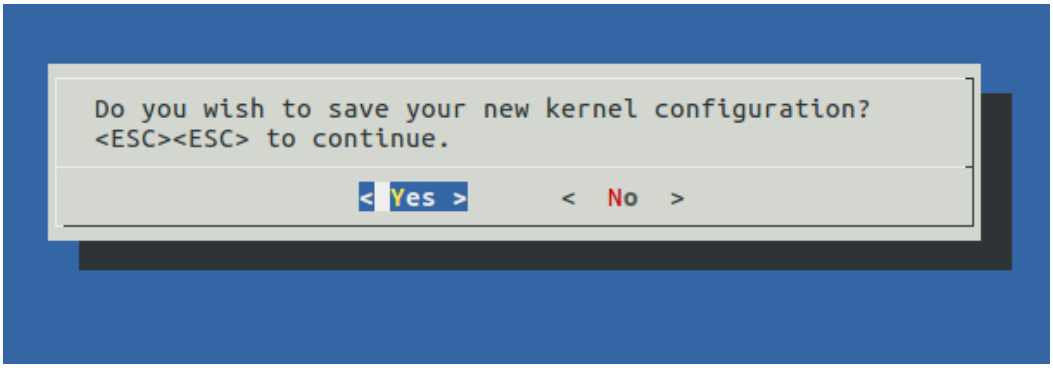
（2）进入 Miscellaneous filesystems：

```
  [ ]   Default to 'data=ordered' in ext3
  [*]   Ext3 extended attributes
  [*]     Ext3 POSIX Access Control Lists
  [*]     Ext3 Security Labels
  < > The Extended 4 (ext4) filesystem
  [ ] JBD (ext3) debugging support
  < > Reiserfs support
  < > JFS filesystem support
  < > XFS filesystem support
  < > GFS2 file system support
  < > OCFS2 file system support
  < > Btrfs filesystem (EXPERIMENTAL) Unstable disk format
  < > NILFS2 file system support (EXPERIMENTAL)
  [*] Dnotify support
  [*] Inotify file change notification support
  [*] Inotify support for userspace
  [ ] Quota support
  <*> Kernel automounter support
  <*> Kernel automounter version 4 support (also supports v3)
  < > FUSE (Filesystem in Userspace) support
      Caches  --->
      CD-ROM/DVD Filesystems  --->
      DOS/FAT/NT Filesystems  --->
      Pseudo filesystems  --->
  [*] Miscellaneous filesystems  --->
  [*] Network File Systems  --->
      Partition Types  --->
  -*- Native language support  --->
  < > Distributed Lock Manager (DLM)  --->
```

（3）勾选 YAFFS2 file system support

```
  --- Miscellaneous filesystems
  < >   ADFS file system support (EXPERIMENTAL)
  < >   Amiga FFS file system support (EXPERIMENTAL)
  < >   eCrypt filesystem layer support (EXPERIMENTAL)
  < >   Apple Macintosh file system support (EXPERIMENTAL)
  < >   Apple Extended HFS file system support
  < >   BeOS file system (BeFS) support (read only) (EXPERIMENTAL)
  < >   BFS file system support (EXPERIMENTAL)
  < >   EFS file system support (read only) (EXPERIMENTAL)
  <*>   YAFFS2 file system support
  -*-     512 byte / page devices
  [ ]       Use older-style on-NAND data format with pageStatus byte (NEW)
  [ ]         Lets Yaffs do its own ECC (NEW)
  -*-     2048 byte (or larger) / page devices
  [*]       Autoselect yaffs2 format (NEW)
  [ ]       Disable YAFFS from doing ECC on tags by default (NEW)
  [ ]       Disable lazy loading (NEW)
  [ ]     Turn off wide tnodes (NEW)
  [ ]     Force chunk erase check (NEW)
  [*]     Cache short names in RAM (NEW)
  [ ]     Empty lost and found on boot (NEW)
  <*>   Journalling Flash File System v2 (JFFS2) support
  (0)     JFFS2 debugging verbosity (0 = quiet, 2 = noisy)
  [*]     JFFS2 write-buffering support
  [ ]       Verify JFFS2 write-buffer reads
  [ ]     JFFS2 summary support (EXPERIMENTAL)
  [ ]     JFFS2 XATTR support (EXPERIMENTAL)
  [ ]     Advanced compression options for JFFS2
  <*>   Compressed ROM file system support (cramfs)
```

(4)保存退出



4.编译：**make zImage**

```
linux-2.6.32# make zImage
```

```
  SYSMAP  System.map
  SYSMAP  .tmp_System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  AS      arch/arm/boot/compressed/head.o
  GZIP    arch/arm/boot/compressed/piggy.gz
  AS      arch/arm/boot/compressed/piggy.o
  CC      arch/arm/boot/compressed/misc.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
root@luowei-thinkpad:/media/luowei/学习/just-for-fun/linux-2.6.32#
```

5.下载到开发板中进行启动（需要同时下载已经做好的文件系统镜像）
正常的启动信息如下：

```
brd: module loaded
S3C24XX NAND Driver, (c) 2004 Simtec Electronics
s3c24xx-nand s3c2440-nand: Tacls=3, 29ns Twrph0=7 69ns, Twrph1=3 29ns
s3c24xx-nand s3c2440-nand: NAND soft ECC
NAND device: Manufacturer ID: 0xec, Chip ID: 0xda (Samsung NAND 256MiB 3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 527 at 0x0000041e0000
Bad eraseblock 1108 at 0x000008a80000
Bad eraseblock 1949 at 0x00000f3a0000
Creating 5 MTD partitions on "NAND 256MiB 3,3V 8-bit":
0x000000000000-0x000000040000 : "supervivi"
uncorrectable error :
0x000000040000-0x000000060000 : "param"
uncorrectable error :
0x000000060000-0x000000560000 : "Kernel"
0x000000560000-0x000040560000 : "root"
mtd: partition "root" extends beyond the end of device "NAND 256MiB 3,3V 8-bit" -- size truncated to 0xfaa0000
ftl_cs: FTL header not found.
0x000000000000-0x000040000000 : "nand"
mtd: partition "nand" extends beyond the end of device "NAND 256MiB 3,3V 8-bit" -- size truncated to 0x10000000
```

```
yaffs: dev is 32505859 name is "mtdblock3"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.3, "mtdblock3"
yaffs: auto selecting yaffs2
block 485 is bad
block 1066 is bad
block 1907 is bad
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
Partially written block 1596 detected
```

```
Partially written block 1590 detected
Partially written block 1590 detected
Partially written block 1590 detected
yaffs_read_super: isCheckpointed 0
VFS: Mounted root (yaffs filesystem) on device 31:3.
Freeing init memory: 128K
hwclock: can't open '/dev/misc/rtc': No such file or directory
[01/Jan/1970:00:00:17 +0000] boa: server version Boa/0.94.13
[01/Jan/1970:00:00:17 +0000] boa: server built Jul 26 2010 at 15:58:29.
[01/Jan/1970:00:00:17 +0000] boa: starting server pid=754, port 80

open device leds: No such file or directory
Try to bring eth0 interface up......ifconfig: SIOCGIFFLAGS: No such device
ifconfig: SIOCSIFHWADDR: No such device
ifconfig: SIOCSIFADDR: No such device
route: SIOCADDRT: No such process
Done

Please press Enter to activate this console.
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
```

至此，内核移植过程完成。可以正常启动登录了，意味着可以在此基础上开发驱动模块，或者开发应用程序了。

注意：内核移植成功后装载其他模块，需要在本次移植后的内核源码目录环境下编译.ko 文件，否则 insmod 会出错：

```
[root@FriendlyARM /drvier-test]# insmod my-buttons.ko
my_buttons: version magic '2.6.32.2-FriendlyARM mod_unload ARMv4 ' should be '2.6.32 mod_unload ARMv4 '
insmod: cannot insert 'my-buttons.ko': invalid module format
```

四、解决 ubuntu16.04-64 位系统不能使用 Minitools 烧录工具的问题

1.运行./MiniTool_x64 发现出错，缺少 libQtWbeKit.so.4

解决方法：sudo apt-get install libqt4-webkit

```
luowei@luowei-thinkpad:~/下载/MiniTools-20140317$ ./MiniTools_x64
./MiniTools_x64: error while loading shared libraries: libQtWebKit.so.4: cannot open shared object file: No such file or directory
luowei@luowei-thinkpad:~/下载/MiniTools-20140317$ sudo apt-get install libqt4-webkit
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列软件包是自动安装的并且现在不需要了：
  linux-headers-4.4.0-101 linux-headers-4.4.0-101-generic linux-headers-4.4.0-96 linux-headers-4.4.0-96-generic linux-headers-4.4.0-97
  linux-headers-4.4.0-97-generic linux-headers-4.4.0-98 linux-headers-4.4.0-98-generic linux-image-4.4.0-101-generic
  linux-image-4.4.0-96-generic linux-image-4.4.0-97-generic linux-image-4.4.0-98-generic linux-image-extra-4.4.0-101-generic
  linux-image-extra-4.4.0-96-generic linux-image-extra-4.4.0-97-generic linux-image-extra-4.4.0-98-generic
使用'sudo apt autoremove'来卸载它(它们)。
将会同时安装下列软件：
  libqt4-opengl libqtwebkit4
下列【新】软件包将被安装：
  libqt4-opengl libqt4-webkit libqtwebkit4
升级了 0 个软件包，新安装了 3 个软件包，要卸载 0 个软件包，有 73 个软件包未被升级。
需要下载 9,342 kB 的归档。
解压缩后会消耗 38.4 MB 的额外空间。
您希望继续执行吗？ [Y/n] Y
获取:1 http://cn.archive.ubuntu.com/ubuntu xenial/main amd64 libqt4-opengl amd64 4:4.8.7+dfsg-5ubuntu2 [301 kB]
获取:2 http://cn.archive.ubuntu.com/ubuntu xenial/universe amd64 libqtwebkit4 amd64 2.3.2-0ubuntu11 [9,034 kB]
获取:3 http://mirrors.163.com/ubuntu precise-security/universe amd64 libqt4-webkit amd64 4:4.8.1-0ubuntu4.9 [8,122 B]
已下载 9,342 kB，耗时 3秒 (2,784 kB/s)
正在选中未选择的软件包 libqt4-opengl:amd64。
(正在读取数据库 ... 系统当前共安装有 364910 个文件和目录。)
正准备解包 .../libqt4-opengl_4%3a4.8.7+dfsg-5ubuntu2_amd64.deb  ...
正在解包 libqt4-opengl:amd64 (4:4.8.7+dfsg-5ubuntu2) ...
正在选中未选择的软件包 libqtwebkit4:amd64。
正准备解包 .../libqtwebkit4_2.3.2-0ubuntu11_amd64.deb  ...
正在解包 libqtwebkit4:amd64 (2.3.2-0ubuntu11) ...
正在选中未选择的软件包 libqt4-webkit。
正准备解包 .../libqt4-webkit_4%3a4.8.1-0ubuntu4.9_amd64.deb  ...
正在解包 libqt4-webkit (4:4.8.1-0ubuntu4.9) ...
正在处理用于 libc-bin (2.23-0ubuntu9) 的触发器 ...
正在设置 libqt4-opengl:amd64 (4:4.8.7+dfsg-5ubuntu2) ...
正在设置 libqtwebkit4:amd64 (2.3.2-0ubuntu11) ...
正在设置 libqt4-webkit (4:4.8.1-0ubuntu4.9) ...
正在处理用于 libc-bin (2.23-0ubuntu9) 的触发器 ...
luowei@luowei-thinkpad:~/下载/MiniTools-20140317$ ./MiniTools_x64
```

2.打开软件发现连不上开发板的问题：

解决方法：必须在 root 模式启动软件下才可以连接上开发板

3.进入 bootloader 下载模式发现没有参考文档中所说的 bootloader 配置菜单
解决方法：需要用 J-link 将 supervivi.bin 文件烧写到 nor flash 中

4.几种内核镜像的区别
Image：内核映像文件;
zImage：映像压缩文件;
uImage：uboot 专用的映像文件，它是在 zImage 之前加上一个长度为 64 字节的“头”，说明这个内核的版本、加载位置、生成时间、大小等信息，其 0x40 之后与 zImage 没有区别。