mini2440.git是该项目是qemu.git项目的一个分支。

描述：三星s3c2440和arm920t两款芯片支持mini2440开发板

主页：<http://code.google.com/p/mini2440/>

作者联系方式： [buserror@gmail.com](mailto:buserror@gmail.com)

URL：

git://repo.or.cz/qemu/mini2440.git

<http://repo.or.cz/qemu/mini2440.git>

Push URL：

ssh://repo.or.cz/qemu/mini2440.git

<https://repo.or.cz/qemu/mini2440.git>

关键词：arm920t dm9000 micro2440 mini2440 qemu s3c2410 s3c2440 s4c2410

# 0 主机环境信息

|  |
| --- |
| Linux version 4.10.0-42-generic (buildd@lgw01-amd64-051) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.5) ) #46~16.04.1-Ubuntu SMP Mon Dec 4 15:57:59 UTC 2017 |

# 1. 交叉编译工具

安装：

sudo apt-get install gcc-arm-linux-gnueabi

这样安装后，交叉编译工具是最新版的，在编译较低内核版本时，发现遇到问题，所以，这儿可以尽量安装较低版本的交叉编译工具，比如4.7。当然，如果已经安装了最新版的交叉编译工具，可以按照后面一节，把版本降低即可解决。

# 2 qemu-mini2440

## 2.1下载qemu for mini2440

git clone git://repo.or.cz/qemu/mini2440.git qemu

如果感觉速度慢，直接打包下载

http://repo.or.cz/w/qemu/mini2440.git/snapshot/HEAD.tar.gz

## 2.2 编译qemu for mini2440

解压后，进入源代码的根目录中，

# ./configure --target-list=arm-softmmu --prefix=/usr/local

# make -j4

## 2.3 编译qemu for mini2440时，遇到的问题

（1）undefined reference to symbol 'timer\_settime@@GLIBC\_2.2

错误内容：

/usr/bin/ld: ../qemu-timer.o: undefined reference to symbol 'timer\_settime@@GLIBC\_2.2'

/usr/bin/ld: note: 'timer\_settime@@GLIBC\_2.2' is defined in DSO /lib/i386-linux-gnu/librt.so.1 so try adding it to the linker command line

/lib/i386-linux-gnu/librt.so.1: could not read symbols: 无效的操作

collect2: 错误： ld 返回 1

make[1]: \*\*\* [qemu-system-arm] 错误 1

make: \*\*\* [subdir-arm-softmmu] 错误 2

搜了半天，后来在一个论坛里找到一个方法，修复了这个问题，但是我目前还没弄明白问题的所在，先把解决方案放这儿备用吧。

论坛的地址是：http://forum.gns3.net/topic6494.html

但是怕地址失效，就把关键的地方简介的说下：

主要是里面提供了一个patch，给configure文件打完补丁之后就解决了，patch里面的内容是：

--- configure.orig 2013-06-13 10:56:41.516276000 +0100

+++ configure 2013-06-13 11:20:26.010368566 +0100

@@ -1362,7 +1362,7 @@

cat > $TMPC <<EOF

#include

#include

-int main(void) { clockid\_t id; return clock\_gettime(id, NULL); }

+int main(void) { struct itimerspec v; timer\_t t; return timer\_gettime (t, &v); }

EOF

rt=no

可以直接把上面的内容拷贝到自己创建的空文件中，文件名命名为configure.patch，但是我觉得其他的名字应该也可以。或者直接下载configure.patch文件，下载地址是：<http://forum.gns3.net/download/file.php?id=1990&sid=2a58777d1298f176a295e8f7baacd630>。然后执行下面的指令

patch configure configure.patch

之后重新configure后再make就没有问题了。

（2）fatal error: SDL.h: 没有那个文件或目录

错误内容：

|  |
| --- |
| The error log from compiling the libSDL test is:  /tmp/qemu-conf--3170-.c:1:17: fatal error: SDL.h: 没有那个文件或目录  compilation terminated.  ERROR: QEMU requires SDL or Cocoa for graphical output  To build QEMU without graphical output configure with --disable-gfx-check  Note that this will disable all output from the virtual graphics card  except through VNC or curses. |

那么就可以使用以下语句修复：

sudo apt-get install libsdl-dev

再使用 make -j4 时， 即可编译成功。

# 3. u-boot for mini2440

## 3.1下载u-boot for mini2440

git clone git://repo.or.cz/w/u-boot-openmoko/mini2440.git uboot

或者打包下载

http://repo.or.cz/w/u-boot-openmoko/mini2440.git/snapshot/HEAD.tar.gz

## 3.2 编译u-boot for mini2440

（注意 采用打包下载的时候这几个包的文件名可能相同，注意区分）解压后，配置Makefile文件，打开Makefile文件，CROSS\_COMPILE变量赋值，即自己所使用的交叉编译工具链，比如我的是arm -linux-gnueabi-，保存退出，输入

# make mini2440\_config

# make -j4

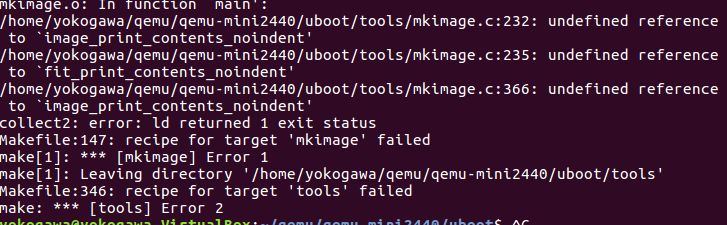
稍等两分钟，即在当前目录下生成名为 u-boot.bin 的文件，注意如果想在之后使用u-boot 的nfs下载文件功能，需要修改代码中的一部分，将net/nfs.c文件中的

NFS\_TIMEOUT = 2UL 修改为 NFS\_TIMEOUT = 20000UL 否则会造成nfs文件下载失败，如果不使用nfs下载功能，不改也可。

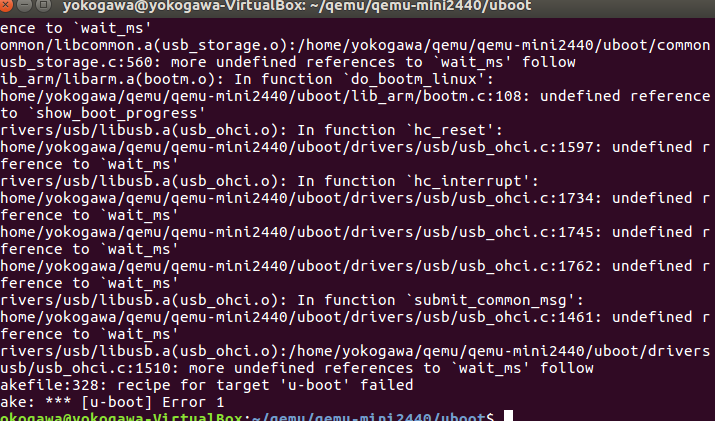
## 3.3 编译uboot时遇到的错误

1.mkimage编译错误，看下面两幅图片：

undefined reference to `image\_print\_contents\_noindent`



undefined reference to `show\_boot\_progress`:

发现上述出现错误的地方都是inline函数，而且声明和定义在不同的文件，且前面都有inline关键字。觉得问题可能出在inline关键字的使用上，于是上网搜索资料后，发现一篇文章，分析的挺好，《[C语言inline详细讲解](http://www.cnblogs.com/xkfz007/articles/2370640.html)》其中一段话，是这样说的：

本文介绍了GCC和C99标准中inline使用上的不同之处。inline属性在使用的时候，要注意以下两点：**inline关键字在GCC参考文档中仅有对其使用在函数定义（Definition）上的描述，而没有提到其是否能用于函数声明（Declare）**。

**从 inline的作用来看，其放置于函数声明中应当也是毫无作用的：inline只会影响函数在translation unit（可以简单理解为C源码文件）内的编译行为，只要超出了这个范围inline属性就没有任何作用了。所以inline关键字不应该出现在函数声明 中，没有任何作用不说，有时还可能造成编译错误（在包含了sys/compiler.h的情况下，声明中出现inline关键字的部分通常无法编译通 过）；**

**inline关键字仅仅是建议编译器做内联展开处理，而不是强制。在gcc编译器中，如果编译优化设置为-O0，即使是inline函数也不会被内联展开，除非设置了强制内联（\_\_attribute\_\_((always\_inline))）属性。**

所以，将上面出错的地方的定义处，把inline关键字去掉，然后重新编译，通过。

# 4. linux kernel for mini2440

## 4.1 下载linux kernel for mini2440

git clone git://repo.or.cz/linux-2.6/mini2440.git kernel

## 4.2 编译linux kernel for mini2440

$ make ARCH=arm mini2440\_defconfig

$ make -j4 ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- uImage

## 4.3 编译linux kernel遇到的问题

问题1：fatal error: linux/compiler-gcc5.h: No such file or directory

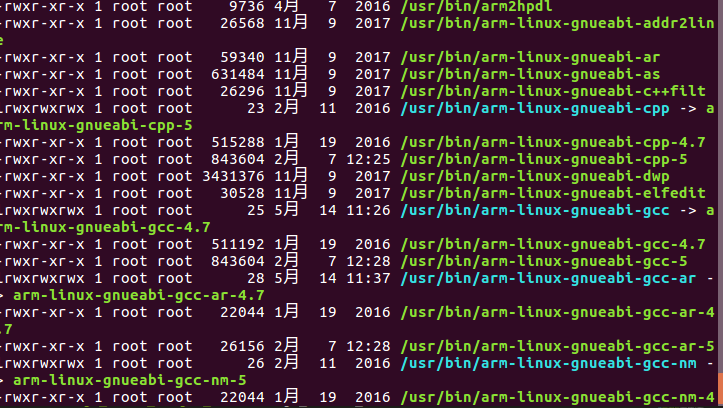
原因：

应该是使用的arm交叉编译工具中需要这个头文件，因为已经安装的arm交叉编译工具版本较高，但是linux内核源代码在编写的时候还没有这个头文件。据此推断，需要降低交叉编译工具的版本。

方法：

|  |
| --- |
| sudo apt install gcc-4.7-arm-linux-gnueabi |

执行完命令后，我们查看主机/usr/bin目录下关于交叉编译工具的信息如下图所示：



我们从图中可以看到，我们实际使用的arm-linux-gnueabi-gcc等工具就是一个符号链接文件，实际指向对应版本的文件，比如，我现在的主机上：

|  |
| --- |
| /usr/bin/arm-linux-gnueabi-gcc -> arm-linux-gnueabi-gcc-4.7 |

出现上面的问题时，我的arm-linux-gnueabi-gcc实际指向的是

/usr/bin/arm-linux-gnueabi-gcc-5，

所以在这里，删除原有的链接文件，新建降低版本后的链接文件，命令如下：

ln -s src dest

比如：

ln -s arm-linux-gnueabi-gcc-4.7 arm-linux-gnueabi-gcc

这样之后，执行：

arm-linux-gnueabi-gcc –version

之后，就可以看到版本信息：

|  |
| --- |
| arm-linux-gnueabi-gcc (Ubuntu/Linaro 4.7.4-3ubuntu12) 4.7.4  Copyright (C) 2012 Free Software Foundation, Inc.  This is free software; see the source for copying conditions. There is NO  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. |

结果：重新编译内核源代码就可以了。

问题2：在Ubuntu 64位环境下编译Liunx内核出现 Can't use 'defined(@array)' 错误的解决办法

内容：

Can't use 'defined(@array)' (Maybe you should just omit the defined()?) at kernel/timeconst.pl line 373.

原因：错误中已经给出了解决方法，就是省略掉defined()就可以了。

结果：重新编译成功。

# 5. rootfs for mini2440

# 6. 启动内核

## 6.1 Nand启动内核

在flashimg文件下，新建一个脚本文件方便启动，文件名

qemu-system-arm-mini2440-nand-start.sh

文件内容

|  |
| --- |
| #!/bin/bash  /**qemu源码所在目录**/mini2440/arm-softmmu/qemu-system-arm -M mini2440 -serial stdio -mtdblock nand.bin -usbdevice mouse |

这里使用绝对路径的原因是因为，我的usr/local/bin目录下已经存在其他版本的qemu-system-arm。

没有挂载根文件系统rootfs的结果如下：

|  |
| --- |
| ini2440-nand-start.sh  mini2440\_init: Boot mode: NAND  S3C: CLK=240 HCLK=240 PCLK=240 UCLK=57  QEMU: ee24c08\_init  DM9000: INIT QEMU MAC : 52:54:00:12:34:56  QEMU mini2440\_reset: loaded default u-boot from NAND  QEMU mini2440\_reset: loaded override u-boot (size 39a00)  S3C: CLK=240 HCLK=60 PCLK=30 UCLK=57  S3C: CLK=240 HCLK=60 PCLK=30 UCLK=48  S3C: CLK=405 HCLK=101 PCLK=50 UCLK=48  U-Boot 1.3.2-mini2440 (May 13 2018 - 22:02:27)  I2C: ready  DRAM: 64 MB  NOR Flash not found. Use hardware switch and 'flinit'  Flash: 0 kB  NAND: Bad block table not found for chip 0  Bad block table not found for chip 0  64 MiB  \*\*\* Warning - bad CRC or NAND, using default environment  USB: S3C2410 USB Deviced  In: serial  Out: serial  Err: serial  MAC: 08:08:11:18:12:27  Hit any key to stop autoboot: 0  MINI2440 # |

上面是Uboot启动结果。启动之后，还需要配置一下（其中mini2440=3tb 是为了使屏幕分辨率成为320\*240，横向显示。默认是240\*320，是竖着的）

|  |
| --- |
| MINI2440 # nboot kernel  MINI2440 #  setenv bootargs root=/dev/mtdblock3 rootfstype=jffs2 console=ttySAC0,115200 mini2440=3tb  MINI2440 # saveenv  MINI2440 # bootm |

启动界面如下，报错是因为我们还没有挂载rootfs。



# 7 Flashimg工具

## 7.1 关于Flashimg

Flashimg是一个强大的工具，是一个由网友FabriceJouhaud 开发的软件，可以很快捷地生成NAND或NOR镜像文件。在不了解flash内部组成和操作原理的情况下，这个软件就可以生成你所想要的大小的镜像文件。

## 7.2 使用flashimg

1. 首先下载flashimg：

git clone git://gitorious.org/flashimg/flashimg.git

2. 在linux系统编译安装：

|  |
| --- |
| $ ./autogen.sh  $ ./configure  $ make  $ sudo make install |

编译成功后，出现一个可执行文件flashimg。

查看版本：

flashimg -v

详细信息：

|  |
| --- |
| flashimg version 0.2 |

要生成NAND或NOR镜像文件，可以先把之前生成的2个文件：u-boot.bin, uImage 拷贝到flashimg（新建）文件夹下，然后把编译flashimg源码目录下的uboot.part也拷贝过来，执行下面的命令，生成NAND或NOR镜像文件：

$ flashimg –s 64M -t nand -f nand.bin -p uboot.part -w boot,u-boot.bin -w kernel,uImage -w root,rootfs.jffs2 -z 512

$ flashimg -s 2M -t nor-f nor.bin -p uboot.part -w boot,u-boot.bin -w kernel,uImage -w root,rootfs.jffs2

其中，

-s 镜像文件的大小

-t 类型指定nand或nor

-f 镜像文件

-p 指定名字，偏移和大小

-w 指定生成镜像的源文件

-z 页大小