搭建嵌入式 ARM Linux 开发环境

2010 年 by 李阳
kmalloc@163.com

1. jtag

1.1. openjtag 编译

一、编译 usb 库

```
#tar xjvf libftd2xx0.4.16.tar.bz2

#cd libftd2xx0.4.16/

#cp libftd2xx.so.0.4.16 /lib

#ln -s /lib/libftd2xx.so.0.4.16 /lib/libftd2xx.so                    //编译 openocd 时需要

#ln -s /lib/libftd2xx.so.0.4.16 /lib/libftd2xx.so.0                  //执行 openocd 时需要

#cd libusb-0.1.12

#./configure

#make

#make install
```

二、编译 openocd

```
#tar xjvf openocd-0.3.1.tar.bz2

#cd openocd-0.3.1

#./configure --enable-parport --enable-parport_ppdev –enable-ft2232_ftd2xx
--enable-usbprog --enable-jlink --with-ftd2xx-linux-
tardir=`pwd`/../libftd2xx0.4.16

#make
```

Note:

编译会出错，提示找不到 API 的引用，因为链接时没有加 -lftd2xx，所以编译时找不到对应的库

解决办法:

可以到 src 目录下手动链接

```
#cd src

#gcc -std=gnu99 -g -O2 -I$PWD/../../libftd2xx0.4.16 -Wall -Wstrict-prototypes
-Wformat-security -Wextra -Wno-unused-parameter -Wbad-function-cast
-Wcast-align -Wredundant-decls -Werror -o openocd main.o
$PWD/../../libftd2xx0.4.16/static_lib/libftd2xx.a.0.4.16  ./.libs/libopenocd.a
-lusb -ldl -lpthread -lftd2xx

#cd ..

#make install
```

三、 配置 USB

```
#cd ..
#cp 50-ftdi.rules  /etc/udev/rules.d/
```

此时插入 openjtag 设备后会有/dev/ttyUSB0 和/dev/ttyUSB1 两个设备文件。

1.2 使用 openocd

1.2.1 运行 openocd:

注：

Openocd 软件运行需要指定对应的配置文件，在此配置文件中有针对我们单板的一些配置。

方法一:

```
#openocd -f <openocd.cfg 目录>/openocd.cfg
```

方法二:

```
#openocd（当前目录下要有 openocd.cfg 文件）
```

1.2.2 通过 telnet 远程输入命令控制 openjtag

openocd 不能直接使用，需要 telnet 连接使用

```
#telnet 127.0.0.1 4444
```

127.0.0.1 是启用 openocd 的主机 ip，4444 是 openocd 的端口号（在 openocd.cfg 里配置），可以远程调试

2. 交叉编译器

2.1. 安装

```
#mkdir -p /usr/local/arm
#tar xjvf gcc-4.3.2.tar.bz2 -C /usr/local/arm/
```

2.2. 修改环境变量

```
#vim ~/.bashrc
```

在文件内添加

```
export PATH=$PATH:/usr/local/arm/4.3.2/bin
```

也可以修改/etc/profile

最后执行新环境变量生效

```
#. ~/.bashrc
```

3. bootloader

3.1. 编译 u-boot

修改配置， vim include/configs/qt2440.h

#make clean

#make distclean

#make qt2440_config

#make

du -sh u-boot.bin (> 237K)

3.2. 烧写 bootloader 到 nandflash

3.2.1 启动 openocd 服务器

```
#cd /tftpboot (下面有对应的 openjtag.cfg)
#openocd
```

3.2.2 烧写 bootloader

登录 openocd 用命令操作 jtag

```
#telnet localhost 4444
```

1. 重启 cpu ，并暂停 cpu，操作时必须先暂停 cpu

```
reset;halt
```

2. 将 CPU Cache 设为 disable 状态

```
arm920t cp15 2 0
step
```

3. 探测 flash，操作 flash 前必须先探测，0 表示第一个 flash

```
nand probe 0
```

4. 擦除 flash

擦除第一个 flash 的第 0 -0x100000 个字节，起始地址和结束地址必须是块大小的倍数

注：现在开发板上 flash 的型号是 K9F2G08UOA，页大小是 2048 字节，64 个页为一个块，块大小是 0x20000 字节（128K）, 我们现在擦 1M 的大小.

```
nand erase 0 0 0x100000
```

5. 确定擦 flash 成功

读第一个 flash 的第 0 -2048 个字节, 并保存在/tftpboot/dump.bin

```
nand dump 0 /tftpboot/2048.nand 0 2048
```

```
#vim /tftpboot/2048.nand
:%!xxd(看 hex)
```

如果为全"1",或全"0"则正确

如果不成功，

```
reset;halt
nand erase 0 0 0x100000
nand dump 0 /tmp/2048.nand 0 2048
```

如果为全"1",或全"0"则正确

6. 烧写 u-boot.bin 到 flash

烧写 u-boot.bin 到 flash 的 0 位置处

```
nand write 0 /tftpboot/u-boot.bin 0
```

7. 读 flash，查看烧写是否成功

```
nand dump 0 /tftpboot/2048.nand 0 2048
```

```
#vim /tftpboot/2048.nand
:%!xxd(看 hex)
```

如果为其他不一样的数字，则证明 u-boot 已经烧进 nand flash

8. 重启

```
reset
```

4. minicom

配置:

```
#minicom -s
```

4.1 端口配置

切换到 minicom 窗口,如果 minicom 窗口无输出，

ls -l /dev/tty*

看一下你的 minicom 端口对了么？

#minicom -s 配置

4.2 串口通信参数配置

波特率: 115200 8N1

硬件控制流: 无

5. 搭建 tftp 服务器：

```
apt-get --force-yes -y  install tftpd-hpa tftp-hpa xinetd

mkdir /tftpboot
chmod 777 /tftpboot
```

把下面的代码复制到 vim /etc/xinetd.d/tftp

```
service tftp
{
    disable = no
    socket_type = dgram
    wait = no
    user = root
    protocol = udp
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot
    log_on_success = PID HOST DURATION
    log_on_failure = HOST
}


sudo /etc/init.d/xinetd restart
sudo /etc/init.d/tftpd-hpa restart
```

测试：

```
touch /tftpboot/aaa
tftp 10.1.0.248(自己的 IP)
get aaa
```

如果没有任何错误，然后按 q 退出看当前目录下有没有 aaa 文件，如果有证明 tftp 服务配置成功

```
-----------------------------------------------
vim /etc/default/tftpd-hpa
TFTP_USERNAME="tftp"
```

```
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="-l -c -s"

------------------------------------------------
```

6. Linux 内核

6.1 配置并编译内核

cd 到内核源码目录(linux-2.6.29)

cp uplooking_config .config

vim Makefile

CROSS_COMPILE=arm-linux-

make menuconfig

make bzImage

生成内核镜像在 arch/arm/boot

6.2 制作 uImage 内核镜像

//使用 mkimage 来将 linux 的 zImage 转换成 uboot 能识别的 image 文件

cd arch/arm/boot

将 u-boot/tools/mkimage 拷贝到/usr/local/bin（你 PATH 中有的目录）

#./mkimage -A arm -O linux -T kernel -C none -a 0x30008000 -e 0x30008040 -n "linux kernel" -d ./zImage uImage

#cp uImage /tftpboot

//将内核通过 tftp 下载到单板的 0x30008000 内存位置处
>tftp 30008000 uImage

//启动内核
>bootm

注： 此时执行 bootm 30008000 启动将失败，是因为没有文件系统导致无法加载启动文件的缘故。

7. 搭建 nfs 服务器

#apt-get --force-yes -y install nfs-kernel-server nfs-client nfs-common portmap

配置：

$ mkdir -p /nfs_rootfs
$ sudo chmod 755 /nfs_rootfs -R
$ sudo vi /etc/exports

/nfs_rootfs *(rw,sync,no_root_squash)

/etc/init.d/portmap start
/etc/init.d/nfs-kernel-server restart

Note:在开发板 ping 一下，如果没有通，关闭防火墙

8. 嵌入式开发的 2 种模式

一、烧写到 *nandflash* 上

kernel:

1.tftp 30000000 uImage
2.nand erase 100000 220000
3.nand write 30000000 100000 220000
4.setenv bootcmd nand read 0x30008000 0x100000 0x220000;bootm 30008000

fs:

1. nand erase 400000 fc00000
2. nfs
3. mount -t yaffs /dev/mtdblock2 /mnt
4. tar xjvf nfsroot-29.tar.bz2
5. mv nfsroot-29/* /mnt

reboot
setenv bootargs noinitrd mem=64M console=ttySAC0 root=/dev/mtdblock2
saveenv

二、网络开发模式

Minicom:

```
setenv bootcmd tftp 30008000 uImage\; bootm\;

setenv bootargs noinitrd console=ttySAC0,115200 mem=64M root=/dev/nfs nfsroot=192.168.0.3:/nfs_rootfs/nfsroot-29 ip=192.168.0.7:192.168.0.3:192.168.0.1:255.255.255.0::eth0:off
```

```
saveenv
```

PC:

```
tar xjvf nfsroot-29.tar.bz2 -C /nfs_rootfs
```

9. 制作根文件系统

```
------------  make the file system image ----------
//total =  bs * count
1. dd if=/dev/zero of=<image file name>  bs=<block size> count=<number>

//insert the file into the loopback device
2. losetup /dev/loop<N>  <image file name>


//format the block device
3. mkfs.<file system format>  /dev/loop<N>

//detach the image file from the loopback device
4. losetup -d /dev/loop<N>


---------- build the file system bin tools (busybox) ----------
//choice  the link type: static link or dymanic link
1. Build Options ---> Build BusyBox as a satic library.....

// configure the compiler prefix
2. Build Options ---> Do you want to build .....

// configure Installation floder
3. Installoation Options --> busybox installation prefix

// choice init process
4. Init Uitilies ---> init

// choice the shell
5. Shells ---> Choose ..... ---> ash

// build
6. make

// install
7. make install


---------- copy the busybox into the image file -------
1. mount -o loop  <image file name>  <mount floder>
```

2. cp -rf <busybox install floder> <mount floder>

3. cd <mount floder>
4. mkdir dev proc sys
5. mknod dev/console c 5 1
6. mknod dev/null c 1 3

if the busybox linked as dynamic linker
cp -rf <complier installation floder>/arm-linux/lib/* lib/

7. umount <mount floder>
. 解压根文件系统到/nfs_rootfs

10. 引导文件系统的3种方式

---------- A. boot from nand flash ----------
1. nand erase <part start addr> <part size>
2. tftp 30000000 <image file name>
3. nand write 30000000 <part start addr> <part size>

configure command line tag for kernel booting
4. setenv bootargs noinitrd mem=64M console=ttySAC0
root=/dev/mtdblock/2
saveenv

download the kernel image and boot it
5. tftp 30008000 uImage
bootm 30008000

---------- done--------

---------- B. boot from init ram disk----------
1. tftp <ram disk start addr> <image file name>
2. setenv bootargs initrd=<ram disk start addr>,<size> mem=64M
console=ttySAC0
root=/dev/ram0
saveenv

download the kernel image and boot it
3. tftp 30008000 uImage
bootm 30008000

---------- done -------

---------- C. boot from network file system (nfs) -------
configure the nfs service of your host
1. ##edit the /etc/exports and append the following:

<shared root file system> *(rw)

##restart the nfs service
/etc/rc.d/init.d/nfs restart

copy the busybox into the nfs service root dir
2. cp -rf <busybox install floder> <nfs service root floder>
cd <nfs service root floder>
mkdir dev proc sys
mknod dev/console c 5 1
mknod dev/null c 1 3

if the busybox linked as dynamic linker
cp -rf <complier installation floder>/arm-linux/lib/* lib/

setup the command line tags on the development board
3. setenv bootargs noinitrd console=ttySAC0,115200 mem=64M
root=/dev/nfs
nfsroot=192.168.0.3:/<nfs service root floder>
ip=192.168.0.2:192.168.0.3:192.168.0.3:255.255.255.0::eth0:off

download the kernel image and boot it
3. tftp 30008000 uImage
bootm 30008000

---------- done -------