嵌入式系统Linux@ARM9专题页 - 你的阳光学习频道

2008-05-21 玩转工具链 -> 宣告失败

根据昨天的计划，开始重新编译最新版的binutils、arm-linux-gcc和glibc。
一个很严重的问题是——我的磁盘空间不足了。直接使用Windows分区不是个好主意——那样会造成所有文件都变成777属性，看起来很不舒服；用loop设备的方法会好一点。创建loop设备并挂载文件系统的方法：

1. 先挂载上Windows分区，然后cd到此分区的某个目录
2. dd if=/dev/zero of=./gccdisk bs=1k count=1M，创建1G的空白文件
   mkfs.ext3 -i 40000 ./gccdisk，在这个文件中创建ext3文件系统
3. sudo mount -o loop ./gccdisk /sunny/gccdisk，挂载loop文件系统。
4. 根据需要用ln -s创建符号连接

编译还需要lex和yacc的支持，Ubuntu源里没有，这个帖子说sudo apt-get install flex bison，果然安装上了。

正式开始编译！

```
$ export TARGET=arm-linux
$ export PREFIX=/sunny/gccdisk
$ export TARGET_PREFIX=/sunny/gccdisk/arm-linux
```

跳过了kernel的menuconfig，因为已经做过了；准备好内核头文件

```
$ cd linux-2.6.25
$ mkdir -p $TARGET_PREFIX/include
$ cp -r include/linux/ $TARGET_PREFIX/include
$ cp -r include/asm-arm/ $TARGET_PREFIX/include/asm
$ cp -r include/asm-generic/ $TARGET_PREFIX/include
```

开始编译binutils

```
$ tar -xzf binutils-2.9.1.tar.gz
$ cd ../binutils-2.9.1
$ ./configure --target=$TARGET --prefix=$PREFIX
```

出现错误：*** BFD does not support target arm-unknown-linux-gnu.
打开bfd/config.bfd一看，对arm只支持这些格式： arm-*-riscix*、arm-*-pe*、arm-*-aout、arm-*-coff，退而求其次，换a.out吧

```
$ export TARGET=arm-linux-aout
$ export TARGET_PREFIX=/sunny/gccdisk/arm-linux-aout
$ mv $PREFIX/arm-linux $PREFIX/arm-linux-aout
$ ./configure --target=$TARGET --prefix=$PREFIX
```

creating Makefile成功

```
$ make
```

提示libiberty/strerror.c有错，与Ubuntu的头文件定义不一致；在#ifdef NEED_sys_errlist前把这个名称#undef掉，然后#else里的sys_errlist类型改成和Ubuntu头文件一致
```
$ make
```
提示libiberty/cplus-dem.c有错，malloc，参考/usr/include/malloc.h改成void *malloc (size_t __size);通过。
```
$ make
```
没看到错误提示，估计是成功了
```
$ make install
```
打开/sunny/gccdisk一看，出现了很多可执行文件，成功
```
$ cd ..
```

开始编译gcc
```
$ mkdir $PREFIX/gcc-4.3.0
$ ln -l $PREFIX/gcc-4.3.0 gcc-4.3.0
```
上面两步纯粹为了解决磁盘空间不足问题，不是必须步骤
```
$ tar -xjf gcc-core-4.3.0.tar.bz2
$ tar -xjf gcc-g++-4.3.0.tar.bz2
```
如果磁盘足够，可以直接使用gcc-4.3.0.tar.bz2；我不需要java、ada之类的，所以就只要C和C++就行了
由于C++编译器依赖于glibc，现在只能先编译C编译器
```
$ cd gcc-4.3.0
$ ./configure --target=$TARGET --prefix=$PREFIX --without-
headers --with-newlib --enable-languages=c
```
还缺东西：Building GCC requires GMP 4.1+ and MPFR 2.3.0+.
```
$ sudo apt-get install libmpfr-dev
```
mpfr的网站打不开，在谷歌快照里看到了“debian package”，Ubuntu的源里也有这个package
```
$ ./configure --target=$TARGET --prefix=$PREFIX --without-
headers --with-newlib --enable-languages=c
$ make
```
这个郁闷了：*** Configuration arm-linux-aout not supported；返回arm-linux目标吧
```
$ export TARGET=arm-linux
$ export TARGET_PREFIX=/sunny/gccdisk/arm-linux
$ ln -s $PREFIX/arm-linux-aout $TARGET_PREFIX
$ ./configure --target=$TARGET --prefix=$PREFIX --without-
headers --with-newlib --enable-languages=c
$ make
```
10分钟后——checking for suffix of object files... configure: error: cannot compute suffix of object files: cannot compile
仔细看看那本Building Embedded Linux System.chm，原来是make目标错了
```
$ make all-gcc
$ make install-gcc
```
迅速成功，/sunny/gccdisk/bin出现了可爱的gcc
```
$ cd ..
```

下一步是glibc

```
$ mkdir $PREFIX/glibc-2.7
$ ln -s $PREFIX/glibc-2.7 glibc-2.7
$ tar -xjf glibc-2.7.tar.gz
$ mkdir build
$ cd build
$ CC=$PREFIX/arm-linux/bin/gcc ../configure --host=$TARGET --
prefix=$PREFIX --enable-add-ons --with-
headers=$TARGET_PREFIX/include
```

好像不行：checking sysdep dirs... configure: error: The arm is not supported.
看到sysdeps/目录里确实没有arm目录
继续搜索，了解到需要glibc-ports-2.7

```
$ cd ..
$ wget http://ftp.gnu.org/gnu/glibc/glibc-ports-2.7.tar.bz2
$ tar -xjf glibc-ports-2.7
$ CC=$PREFIX/arm-linux/bin/gcc ../configure --host=$TARGET --
prefix=$PREFIX --enable-add-ons=../glibc-ports-2.7 --with-
headers=$TARGET_PREFIX/include
```

编译失败，config.log里说ccl找不到；我在$PREFIX/libexec/gcc/arm-linux/4.3.0找到了ccl这个文件，加入path里

```
$ PATH=$PREFIX/libexec/gcc/arm-linux/4.3.0:$PATH CC=$PREFIX/arm-
linux/bin/gcc AR=$PREFIX/bin/arm-linux-aout-ar
RANLIB=$PREFIX/bin/arm-linux-aout-ranlib AS=$PREFIX/bin/arm-
linux-aout-as LD=$PREFIX/bin/arm-linux-aout-ld ../configure --
host=$TARGET --prefix=$PREFIX --enable-add-ons=../glibc-ports-
2.7 --with-headers=$TARGET_PREFIX/include
```

还是失败，config.log说Error: too many memory references for `mov'，看起来是x86汇编……

```
$ cd ../..
```

暂时没有解决方案，试试uClibc吧

```
$ export CROSS=$PREFIX/bin/arm-linux-
$ ln -s $CROSS'aout-ar' $CROSS'ar'
$ ln -s $CROSS'aout-as' $CROSS'as'
$ ln -s $CROSS'aout-ld' $CROSS'ld'
$ ln -s $CROSS'aout-objcopy' $CROSS'objcopy'
$ ln -s $CROSS'aout-objdump' $CROSS'objdump'
$ ln -s $CROSS'aout-ranlib' $CROSS'ranlib'
$ ln -s $CROSS'aout-strip' $CROSS'strip'
```

先用符号链接统一一下“arm-linux-aout”与“arm-linux”的区别（binutils留下的怪事～）

```
$ tar -xjf uClibc-0.9.29.tar.bz2
$ cd uClibc-0.9.29
$ make CROSS=$CROSS defconfig
$ make CROSS=$CROSS menuconfig
```

修改arm、arm920t、little endian、kernel header location、cross等

```
$ make
```

提示错误：Error: Unknown pseudo-op: `.section'

看起来还是我的编译器有问题！！！！！

体验了编译工具链之后，到这里无法继续。这个网页指出：“Linux 2.6.14 compiles with the 3.4.1 tool chain. It failed with 3.3.2.”，也就是说最新版本的编译器有时候就是会出奇怪问题——特别是我把elf、aout混合使用，天知道会发生什么……

下载了一个别人编译好的编译器
修改kernel、BusyBox等的Makefile，指向新的编译器。make clean、make。
但是——shell还是不理我，呜呜呜……
（另外发现，用下载来的新编译器产生的helloworld，file命令还是看到for GNU/Linux 2.4.3）

时间不多了，我决定尝试另一个版本：`linux-2.4.36.4`。
首先修改顶层Makefile的ARCH、CROSS_COMPILE；居然不支持make defconfig，且make menuconfig会出错……只好老老实实make config，后面也可以vi .config修改少量配置。
make dep也会出错，arch/arm/Makefile里的-mshort-load-bytes删掉即可。

```
$ make
```

- `blkpg.c:252: error: asm-specifier for variable `__r1' conflicts with asm clobber list`
  `rd.c:306: error: asm-specifier for variable `__r1' conflicts with asm clobber list`
  `loop.c:903: error: asm-specifier for variable `__r1' conflicts with asm clobber list`
  不是64位，BLKGETSIZE64一般也不会用到，注释掉
- `/sunny/linux-2.4.36.4/include/asm/keyboard.h:68:31: asm/arch/keyboard.h: No such file or directory`
  SkyEye虚拟开发板上不需要键盘，所以touch include/asm/arch/keyboard.h创建一个空文件
- `filemap.c:1948: error: asm-specifier for variable `__r1' conflicts with asm clobber list`
  这次是sys_sendfile64函数，怎么又是64位的东西？
- `core.c:176: error: `MACH_TYPE_AT91RM9200' undeclared here (not in a function)`
  别处都是MACH_TYPE_AT91RM9200DK，这儿却丢了DK二字，在include/asm/mach-types.h定义一下吧
- `cc1: error: invalid option `no-fpu'`
  arch/arm/Makefile，删去-mno-fpu

Generating arch/arm/vmlinux.lds——终于通过了编译阶段，到了ld；错误更多了，都是undefined reference……

又是内核的问题，无法解决，放弃……用原来的那个编译器尝试，问题依旧
我不想继续这个项目，下面开始整理代码、写报告

2008-05-20 shell不理人，解决不了吗？

继续分析了昨天最后遇到的kernel_execve函数，汇编最后一行是b ret_to_user。这个符号定义于arch/arm/kernel/entry-common.S，这篇文章指出它是用于系统调用后从内核态返回用户态、并包含进程调度功能。从昨天的一部分输出看，为了执行/etc/init.d/rcS里的shell脚本，内核创建了sh进程；rcS里写个mount命令，内核就会创建mount进程——这意味着，ret_to_user没有问题，用户态程序能够获得控制权。
我估计，存在的问题是，shell的输出没有到达ttyS0这个串口上、而是输出到了其他地方。BusyBox的FAQ指出，写一个静态链接的hello world作为init，如果看不到打印的消息，就先不要责怪BusyBox有问题；我根据提示尝试了，确实没能看到hello world。

又一次开始狂翻邮件列表

- 这个帖子的问题和我很像
  `In short, if you use buildroot , you will not be able to see any login prompt!`
  解决方法是：
  `Make sure your /dev directory contain reasonable files such as /dev/console /dev/stdin /dev/sdtout`
  我没有用buildroot也没有用uClibc，但是我仍然根据提示建立了/dev/stdin、stdout、stderr三个符号链接。
  没有解决我的问题。
- 看见了“maxim's patch”，搜索之——是这个网页上提供的内核补丁（arm linux已经集成于内核、不需要自己patch，但是at91开发板还是需要对内核进行小小的修改）。patch -p0 -i 2.6.25-at91.pathc，打上补丁（patch的用法看这里）。
  没有解决我的问题。

用file命令发现我用的交叉gcc、glibc都是for GNU/Linux 2.4.3，难道是这个问题？下载了binutils、gcc、glibc源码，打算重新编译。comic上有本好书（仅限校内访问，用xchm打开观看）。

2008-05-19 shell，你能听到我吗？

事实上，15日最终的配置，init就是不能启动的：把libc、libcrypt复制进去还不够，还需要ld才能这个库，init才能启动。
于是——内核执行到达run_init_process("/sbin/init")后（在内核代码中插入printk可以看出），就不再输出任何东西了。
一步步阅读内核源码、在适当位置插入printk，发现BusyBox的init要读取/etc/inittab、并执行/etc/init.d/rcS，而我的initrd里没有这两个文件；参考网上资料建立之。
现在的问题是，从插入的printk输出中已经看出，内核已经能正确读取initrd的内容、开始加载进程。从init/main.c的run_init_process调用了arch/arm/kernel/sys_arm.c的kernel_execve函数，进入里面的汇编代码后就没出来；暂不理解这段汇编的含义。我看不到shell的任何输出。

2008-05-15 直接引导，可以吗？

查看了Linux kernel的arch/arm/mach-at91/Makefile.boot文件：

```
       zreladdr-y    := 0x20008000
    params_phys-y    := 0x20000100
    initrd_phys-y    := 0x20410000
```

莫非这就是内核应该载入的物理地址、以及默认寻找initrd的物理地址？
把uImage的载入的地址修改成上述地址，但还是在uncompress完毕后停住

```
$ ../u-boot-1.3.2/tools/mkimage -A arm -O linux -T kernel -C
none -a 20008000 -e 20008000 -n linux-2.6.25 -d ./zImage uImage
$ ../u-boot-1.3.2/tools/mkimage -A arm -O linux -T ramdisk -C
none -a 20410000 -e 20410000 -n linux-2.6.25-initrd -d ./initrd
initrd.u
```

我想到，能否不用U-Boot，直接引导Linux呢？
写了个SHELL脚本myimage.sh，创建一个32M的文件准备装载在0x20000000，然后把zImage或Image、initrd写在合适的位置。

```
[    0.000000] Linux version 2.6.25yoursunny (sunny@hardy) (gcc
version 3.4.5) #29 Thu May 15 21:22:46 CST 2008
[    0.000000] CPU: ARM920T [41009200] revision 0
(ARMvundefined/unknown), cr=00003177
[    0.000000] Machine: Atmel AT91RM9200-DK
```

内核参数只能在内核的menuconfig里指定，因为没有U-Boot给它传参数了：

```
[    0.007812] Kernel command line: root=/dev/ram
initrd=0x20410000,0x00100000 rw console=ttyS0 mem=32M
init=/sbin/init
```

但是——

```
[   10.007812] Failed to execute /sbin/init.  Attempting
defaults...
[   10.039062] Kernel panic - not syncing: No init found.  Try
passing init= option to kernel.
```

在这里看到，BusyBox可能有外部依赖，用readelf一看，果然：

```
$ /usr/local/arm/bin/arm-unknown-linux-gnu-readelf -a
~/study/IS222/busybox-1.9.2/busybox | grep Shared
 0x00000001 (NEEDED)                     Shared library:
[libcrypt.so.1]
 0x00000001 (NEEDED)                     Shared library:
[libm.so.6]
 0x00000001 (NEEDED)                     Shared library:
[libc.so.6]
```

修改initrd.sh脚本，把这3个lib也cp进去、且内核menuconfig支持so模块，/sbin/init就能够启动了
但是——

```
[   18.546875] attempt to access beyond end of device
```

```
[   18.554687] ram0: rw=0, want=3408159018, limit=8192
[   18.562500] Buffer I/O error on device ram0, logical block
3851563156
```
发现是内核启动参数initrd忘了改，改为initrd=0x20410000,0x00400000，那个No init found的老毛病又来了。sigh~

2008-05-14 U-Boot引导Linux

盲目中试了很多次go c0000000（我在这个地址载入了linux内核的zImage），总是出错、SKYEYE提示no bank。

看了资料，说U-Boot应该使用uImage格式、而不是zImage或vmlinux或Image，用mkimage命令可以制作zImage。
还写了个SHELL脚本initrd.sh，自动创建initrd：dd一个1M的空文件，mount成loopback设备，在里面mknod一些dev，将busybox安装进去，然后umount。
运行！在uncompress kernel完毕booting the kernel时，就停住了，什么反应也没有。不知如何解决……

今天查看的主要资料：

- http://www.linux4sam.org/twiki/bin/view/Linux4SAM/WebHome
- http://www.denx.de/wiki/view/DULG/RootFileSystemOnARamdisk

2008-05-08 编译Linux Kernel、U-Boot、BusyBox

主机重装了，Ubuntu Hardy、gnome，用alternate光盘安装的，而且专门做了ext3的分区，不再使用NTFS+Wubi。

今天编译了2.6.25版本的Linux内核，用的编译器是arm-gcc-3.4.5 & glibc2.3.6；是的，我总是热衷于最新的平台。
操作很简单的，kernel解压后在Documentation/arm/README可以找到说明：

1. 打开顶层的Makefile，找到ARCH，改成ARCH=arm；下面的CROSS_COMPILE写上编译器的路径，比如CROSS_COMPILE=/usr/local/arm/bin/arm-unknown-linux-gnu-，就是gcc可执行文件的绝对路径去掉最后的gcc
2. make config，出现数千个选项，我找个东西压住键盘上的ENTER键，然后去喝杯茶，几分钟后接受了所有默认选择
3. make zImage，再喝杯茶，几分钟后完毕，arch/arm/boot/zImage就是成品，1382436字节，压缩前的Image为2777632字节

还有u-boot-1.3.2，README有说明，同样很简单：

1. 打开顶层的Makefile，找到对应arm平台的CROSS_COMPILE，写上编译器的路径
2. make at91rm9200dk_config，调用了AT91RM9200开发板的配置

3. make all，几分钟后完毕，u-boot.bin或u-boot就是成品，分别为96724字节、324517字节

busybox-1.9.2

1. 打开顶层的Makefile，ARCH=arm、CROSS_COMPILE=编译器的路径
2. make defconfig，默认配置
3. make，成品busybox，753072字节

目前只有U-Boot能运行，根据SkyEye的wiki文档一次成功

2008-04-04 我的选择

我选定了开发环境和目标：主机Ubuntu Desktop 7.10，模拟器skyeye-1.2.4，目标开发板Atmel AT91RM9200，Bootloader u-boot-1.3.2 ，内核linux-2.6.24.4
其他找到的资源：交叉编译环境建立
