嵌入式操作系统

4 SkyEye简介

陈香兰（xlanchen@ustc.edu.cn）

计算机应用教研室@计算机学院
嵌入式系统实验室@苏州研究院
中国科学技术大学
Fall 2014

November 28, 2014

Outline

Outline

SkyEye简介

- SkyEye is an Open Source Software Project (GPL Licence).
  - Origin from GDB/Armulator,
  - 在Sourceforge上可以获得Skyeye的相关代码
- SkyEye的起源和发展
  - 陈渝： 做一个用软件实现的嵌入式开发硬件模拟器，可以在模拟器上运行各种操作系统，这样就可以在没有开发板的情况下学习和研究操作系统
- SkyEye的目标：
  to provide an integrated simulation environment in Linux and Windows, simulates/emulates typical Embedded Computer Systems

SkyEye简介

- Now the following OS and system softwares can run in SkyEye:
  - uC/OS-II-2.x with network support
  - uClinux based on Linux2.4.x/2.6.x with Network/LCD/TouchScreen/Flash Mem support
  - ARM Linux 2.4.x/2.6.x with Network/LCD/TouchScreen/Flash Mem support
  - Nucleus
  - Rtems
  - Ecos
  - lwIP on uC/OS-II
  - applications on uC/OSII, uClinux,ARM Linux

可对上述软件系统进行源码级的分析、调试和测试。

SkyEye简介

Processor List supported by SkyEye

Processor name	*Core*	*Architecture*	*Current status*	*Running OS*
S3C4510	ARM7TDMI	ARM	stable	uClinux
S3C44B0	ARM7TDMI	ARM	stable	uClinux
AT91	ARM7TDMI	ARM	stable	uClinux
S3C3410	ARM7TDMI	ARM	stable	uClinux
LPC2210	ARM7TDMI	ARM	not finished	uClinux
EP7312	ATM720T	ARM	stable	linux
Ep9312	ARM9	ARM	stable	linux
AT91RM9200	ARM9	ARM	stable	linux
S3C2410	ARM9	ARM	stable	linux
CS89712	ARM9	ARM	stable	linux
SA1100	StrongArm	ARM	stable	linux
PXA25x	XSCALE	ARM	stable	linux
PXA27x	XSCALE	ARM	stable	linux
bf533	bf53x	Blackfin	testing	uClinux
bf537	bf53x	Blackfin	testing	uClinux
CF5249		Coldfire	stable	uClinux
CF5272		Coldfire	testing	uClinux
Au1100	R4K	MIPS	not finished	linux
MPC8560	E500	PowerPC	testing	linux
MPC8572	E500	PowerPC	testing	linux
leon2	Sparc v8	Sparc	testing	RTEMS

SkyEye模拟硬件介绍

- Now the following hardwares can be simulated by SkyEye:
  - CPU CORE: ARM7TDMI, ARM720T,StrongARM, XScale, Blackfin
  - APPLICATION CPU: Atmel AT91X40/AT91RM9200,Cirrus CIRRUS LOGIC EP7312/EP9312 CS89712,Intel SA1100/SA1110, Intel PXA 25x/27x, Samsung 4510B/44B0/2410/2440 , Sharp LH7xxxx, NS9750, Philips LPC22xx, BF533
  - MEMORY: RAM, ROM, Flash
  - Peripheral: Timer, UART,NIC chip,LCD, TouchScreen, etc.

SkyEye模拟硬件介绍

存储器管理单元和缓存单元

- MMU

  Memory Management Unit，
  存储器管理单元，
  是用来管理虚拟内存系统的硬件。

  - MMU的两个主要功能是：
    1. 将虚地址转换成物理地址；
    2. 控制存储器的存取权限。
  - MMU关掉时，虚地址直接输出到物理地址总线

SkyEye模拟硬件介绍

存储器管理单元和缓存单元

- TLB，
  Translation Lookaside Buffers
  在MMU中，存放从虚拟地址到物理地址的匹配表
  - 保存的内容包括：
    虚址及其对应的物理地址，权限，域和映射类型。
  - 当CPU对一虚拟地址进行存取时，
    首先搜索TLB表以查找对应的物理 地址等信息，
    如果没有查到，则进行查找translation table，称为Translation Table Walk（简称TTW）。
    经过TTW过程后，将查到的信息保存到TLB。然后根据TLB表项的物理地址进行读写。

SkyEye模拟硬件介绍

存储器管理单元和缓存单元

- CACHE，缓存单元
  主要用于缓存内存中的数据，其读写速度远快于内存的读写速度，所以可以提高CPU的内存数据的访问效率。
- write/read buffer硬件单元
  write/read buffer硬件单元的作用与CACHE的作用类似。
- MMU、CACHE、write/read buffer一般是高性能CPU的重要组成部分，且不同类型CPU的MMU、CACHE、write/read buffer的逻辑行为也有一定的差异。为了支持模拟多种类型CPU的MMU/CACHE，SkyEye包含了一个通用的MMU/CACHE模拟实现。通过对一些参数的调整可以支持模拟多种类型的MMU/CACHE物理结构和逻辑行为。

SkyEye模拟硬件介绍

网络芯片

- 目前SkyEye模拟了网络芯片8019AS，
- 其特点是：NE2000兼容，内建16KRAM缓冲区，10MB传输速率。
- 虽然目前模拟的开发板上不一定有网络芯片8019AS，但可以在模拟的开发板上加上网络芯片8019AS 的模拟。

Outline

Outline

SkyEye的安装

1. Linux操作系统
   - 当前使用的主机操作系统是ubuntu-14.04或者Kubuntu-14.04
   - ubuntu-14.04自带的编译器是gcc-4.8.2
2. Windows+MingW或cygwin
   - 请自行摸索

- 在Kubuntu上安装SkyEye有两种安装方法
  1. 直接安装二进制代码

     ```
     sudo apt-get install skyeye
     ```

  2. 下载源代码，编译并安装

在ubuntu-14.04上直接安装skyeye

- 在ubuntu-14.04中，使用命令

$$\text{sudo apt} - \text{get install skyeye}$$

  安装skyeye-1.2.5，版本有些老。
- 使用

$$\text{sky} < \text{tab} >$$

  可以出现完整的skyeye命令

skyeye -h

```
SkyEye 1.2.5
Bug report: skyeye-developer@lists.gro.clinux.org
Usage: skyeye [options] -e program [program args]
Default mode is STANDALONE mode
————————————————————————————
Options:
-e exec-file the (ELF executable format)kernel file name.
-l load_address,load_address_mask Load ELF file to another address, not its entry.
-b specify the data type is big endian when non ”-e” option.
-d in GDB Server mode (can be connected by GDB).
-c config-file the skyeye configure file name.
-h The SkyEye command options, and ARCHs and CPUs simulated.
```

在ubuntu-14.04上直接安装skyeye

```
————— Architectures and CPUs simulated by SkyEye—————-
——— ARM architectures ———
at91
lpc
s3c4510b
s3c44b0x
s3c44b0
s3c3410x
ep7312
lh79520
ep9312
cs89712
sa1100
pxa_lubbock
pxa_mainstone
at91rm92
s3c2410x
s3c2440
sharp_lh7a400
ns9750
——— BlackFin architectures —-
bf533
bf537
```

Outline

试用skyeye-1.2.5 I

- 下载skyeye-1.2.5配套的测试包
  - skyeye-testsuite-1.2.5.tar.bz2
- 解压缩

```
tar -jvxf skyeye-testsuite-1.2.5.tar.bz2
```

- 进入测试包目录

```
cd skyeye-testsuite-1.2.5
```

1. 试运行armlinux

```
cd linux
cd ep7312
cd 2.6.x
skyeye -e vmlinux -c skyeye.conf
```

试用skyeye-1.2.5 II

2. 试运行uClinux

```
cd uClinux
cd at91
cd uclinux_cs8900a
skyeye -e linux -c skyeye.conf
```

- 注意：
  运行testsuite下的例子时，有些可能会出错。

Outline

下载skyeye源代码，编译并安装

- 最新版本情况
  - skyeye-1.3.4_rc1.tar.gz
  - testsuite-1.3.4_rc1.tar.gz
- 解压缩后进入skyeye-1.3.4的目录，根据INSTALL文件
  - ./configure
  - make lib
  - make install_lib
  - make
- 编译skyeye过程中会出现很多错误，提示需要安装一些软件

Outline

Outline

1、准备交叉编译环境

- 对于μCLinux，要使用arm-elf-工具链
- 下载arm-elf-tools-20030314.sh
  - 运行

    ./arm-elf-tools-20030314.sh

    安装
  - 使用

    arm-elf-<tab>

    查看是否安装成功

2、准备hello

hello.c循环输出“HelloWorld!”

```
#include <stdio.h>
int main(void){
    while(1)
        printf(" HelloWorld!\n" );
        return 0;
}
```

- 交叉编译hello

```
arm-elf-gcc -elf2flt -o hello hello.c
```

- 查看hello的文件类型信息

```
file hello
hello: BFLT executable - version 4 ram
```

3、添加hello到μCLinux的根文件系统中

- 挂载

```
mkdir romfs_dir
sudo mount -o loop romfs.img romfs_dir
```

```
sudo mount -o loop romfs.img romfs_dir/
...
mount: warning: romfs_dir/ seems to be mounted read-only.
```

- 直接拷贝hello到romfs_dir？报错！！WHY？

```
sudo cp hello romfs_dir/bin/
cp: 无法创建普通文件”romfs_dir/bin/hello”: 只读文件系统
```

3、添加hello到μCLinux的根文件系统中

- 复制根文件系统到一个新的目录中

```
sudo cp -r romfs_dir/* new_romfs/
```

- 拷贝hello

```
sudo cp hello new_romfs/bin
```

- 生成新的romfs映像

```
sudo genromfs -f romfs_new.img -d new_romfs/
```

4、建立新的skyeye配置文件

- 复制skyeye.conf为skyeye_new.conf

cp skyeye.conf skyey_new.conf

- 修改skyeye_new.cong，使之使用新的romfs映像romfs_new.img

```
#skyeye config file sample
cpu: arm7tdmi
mach: at91
mem_bank: map=M, type=RW, addr=0x00000000, size=0x00004000
mem_bank: map=M, type=RW, addr=0x01000000, size=0x00400000
mem_bank: map=M, type=R, addr=0x01400000, size=0x00400000, file=./romfs_new.img
mem_bank: map=M, type=RW, addr=0x02000000, size=0x00400000
mem_bank: map=M, type=RW, addr=0x02400000, size=0x00008000
mem_bank: map=M, type=RW, addr=0x04000000, size=0x00400000
mem_bank: map=I, type=RW, addr=0xf0000000, size=0x10000000
#set nic info
#net: type=cs8900a, base=0xfffa0000, size=0x20,int=16, mac=0:4:3:2:1:f,
ethmod=tuntap, hostip=10.0.0.1
net: type=cs8900a, ethmod=tuntap, hostip=10.0.0.1
#dbct: state=on
```

5、运行

- 运行skyeye

  skyeye -c skyeye.conf -e linux

- 见到命令提示符/>后，运行

  bin/hello

  将会进入“HelloWorld!”死循环。

Outline

1、建立交叉编译环境

- 下载交叉编译工具
  - ftp://ftp.arm.linux.org.uk/pub/armlinux/toolchain
  - cross-2.95.3.tar.bz2
- 解压缩到/usr/local/arm目录下或当前工作目录下

  tar -jxf cross-2.95.3.tar.bz2
- 设置执行路径，在.bashrc中添加交叉编译器的bin目录

  exprot PATH = $PATH : /usr/local/arm/2.95.3/bin
- 然后，退出控制台，重新启动控制台
  目的：使得更新后的.bashrc文件起作用

1、建立交叉编译环境

- 检查是否建立好交叉编译环境

arm-linux- < tab >

arm-linux-<tab>

arm-linux-addr2line	arm-linux-gasp	arm-linux-protoize
arm-linux-ar	arm-linux-gcc	arm-linux-ranlib
arm-linux-as	arm-linux-gcj	arm-linux-readelf
arm-linux-c++	arm-linux-ld	arm-linux-size
arm-linux-c++filt	arm-linux-nm	arm-linux-strings
arm-linux-g++	arm-linux-objcopy	arm-linux-strip
arm-linux-g77	arm-linux-objdump	arm-linux-unprotoize

1、建立交叉编译环境

- 试运行arm-linux-gcc，查看其版本：

```
arm-linux-gcc -v
```

```
arm-linux-gcc -v
Reading specs from
/usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3/specs gcc
version 2.95.3 20010315 (release)
```

2、准备hello

hello.c循环输出“HelloWorld!”

```
#include <stdio.h>
int main(void){
    while(1)
        printf(" HelloWorld!\n" );
        return 0;
}
```

- 交叉编译，选择静态编译

$$arm-linux-gcc \ -static \ -o \ hello \ hello.c$$

file hello

```
hello: ELF 32-bit LSB executable, ARM, version 1, statically
linked, for GNU/Linux 2.0.0, not stripped
```

3、将hello加入到根文件系统中

- 选择skyeye-testsuite-1.2.5/linux/ep7312/2.6.x
- 查看所使用的根文件系统是哪一个

vi skyeye.conf

```
cpu: arm720t
mach: ep7312
mem_bank: map=M, type=RW, addr=0x00000000, size=0x00400000
mem_bank: map=I, type=RW, addr=0x80000000, size=0x00010000
mem_bank: map=M, type=RW, addr=0xc0000000, size=0x00200000
mem_bank: map=M, type=RW, addr=0xc0200000, size=0x00600000, file=./initrd_old.img
mem_bank: map=M, type=RW, addr=0xc0800000, size=0x00800000
#lcd:type=ep7312,mod=gtk

#dbct:state=on
```

- 挂载根文件系统到rootfs上
  - mkdir rootfs
  - sudo mount -o loop initrd_old.img rootfs/

3、将hello加入到根文件系统中

- 拷贝hello到根文件系统中：报错！！WHY？

```
sudo cp hello rootfs/bin
```

```
sudo cp hello rootfs/bin
cp: 写入”rootfs/bin/hello” 出错: 设备上没有空间
cp: 扩展”rootfs/bin/hello” 失败: 设备上没有空间
```

  - 查看initrd_old.img的文件大小信息，发现其大小为6MB
  - 查看skyeye.conf文件中，加载initrd_old.img的存储空间大小，也是6MB
  - 怎么办？

- 改变映像的大小？改变skyeye.conf中存储空间大小？
  一种可能的解决方法：删除旧的映像中目前用不到的内容

3、将hello加入到根文件系统中

1. 建立一个新的映像文件，并格式化为ext2文件系统

```
dd of=myinitrd6M.img if=/dev/zero bs=2048 count=3072

mke2fs -F -v -m0 myinitrd6M.img
```

2. 将rootfs的内容拷贝到一个temp目录中

```
sudo umount rootfs/
sudo mount -o loop initrd_old.img rootfs/
mkdir temp
sudo cp -r rootfs/* temp/
sudo umount rootfs
```

3. 删除temp中usr/local/lib目录下的minigui目录（暂时用不到）

```
sudo rm -r usr/local/lib/minigui
```

3、将hello加入到根文件系统中

4. 复制temp到新的映像文件中，复制hello到新的映像文件中

```
mkdir rootfs_new
sudo mount -o loop myinitrd6M.img rootfs_new/
sudo cp -r temp/* rootfs_new/
sudo cp hello rootfs_new/bin
sudo umount rootfs_new
```

完整的解决方案后面再介绍

4、建立新的skyeye配置文件

- 复制skyeye.conf到skyeye_new.conf
- 修改skyeye_new.cong，使其使用新的映像文件

```
cpu: arm720t
mach: ep7312
mem_bank: map=M, type=RW, addr=0x00000000, size=0x00400000
mem_bank: map=I, type=RW, addr=0x80000000, size=0x00010000
mem_bank: map=M, type=RW, addr=0xc0000000, size=0x00200000
mem_bank: map=M, type=RW, addr=0xc0200000, size=0x00600000, file=./myinitrd6M.img
mem_bank: map=M, type=RW, addr=0xc0800000, size=0x00800000
#lcd:type=ep7312,mod=gtk
#dbct:state=on
```

5、运行

```
skyeye -e vmlinux -c skyeye_new.conf
```

```
Welcome to
     _      _____       __   __     _       _
    / \    /  __ \     /  \_/  \   | |     |_|
   / _ \   | |  | |   / /\   /\ \  | |
  / /_\ \  | |__| |  / /  \_/  \ \ | |      _ ____   _    _  _  _
 / /___\ \ | |__\ \ | |       | || |__    | |  _ \| | | |\ \/ /
/_/     \_\| |   \_\|_|       |_||____|   |_| |_| | |_| |/    \
ARMLinux for Skyeye
For further information check:
http://hpclab.cs.tsinghua.edu.cn/~skyeye/

Command: #/bin/demobmp
Execution Finished, Exiting
Command: /bin/sh

Sash command shell (version 1.1.1)
/>
```

运行hello

bin/hello

进入”HelloWorld!“死循环。

Outline

Outline

1、准备交叉编译环境 I

- 直接安装

```
sudo apt-get install gcc-arm-linux-gnueabi
arm-linux-gnueabi-<tab>
```

arm-linux-gnueabi-<tab>

```
arm-linux-gnueabi-addr2line          arm-linux-gnueabi-gcov-4.7
arm-linux-gnueabi-ar                 arm-linux-gnueabi-gprof
arm-linux-gnueabi-as                 arm-linux-gnueabi-ld
arm-linux-gnueabi-c++filt            arm-linux-gnueabi-ld.bfd
arm-linux-gnueabi-cpp                arm-linux-gnueabi-ld.gold
arm-linux-gnueabi-cpp-4.7            arm-linux-gnueabi-nm
arm-linux-gnueabi-dwp                arm-linux-gnueabi-objcopy
arm-linux-gnueabi-elfedit            arm-linux-gnueabi-objdump
arm-linux-gnueabi-gcc                arm-linux-gnueabi-ranlib
arm-linux-gnueabi-gcc-4.7            arm-linux-gnueabi-readelf
arm-linux-gnueabi-gcc-ar-4.7         arm-linux-gnueabi-size
arm-linux-gnueabi-gcc-nm-4.7         arm-linux-gnueabi-strings
arm-linux-gnueabi-gcc-ranlib-4.7     arm-linux-gnueabi-strip
arm-linux-gnueabi-gcov
```

1、准备交叉编译环境 II

- 或者
  下载arm-linux-tools-20061213.tar.gz安装
  - 在主机的系统根目录下或在当前目录下解压缩
    ```
    sudo tar zvxf arm-linux-tools-20061213.tar.gz
    ```
  - 查看安装是否成功

./usr/local/bin/arm-linux-<tab>

```
arm-linux-addr2line          arm-linux-g77          arm-linux-jv-scan
arm-linux-addr2name.awk      arm-linux-gcc          arm-linux-ld
arm-linux-ar                 arm-linux-gcc-3.4.4     arm-linux-ld.real
arm-linux-arm-linux-gcjh     arm-linux-gccbug       arm-linux-nm
arm-linux-as                 arm-linux-gcj          arm-linux-objcopy
arm-linux-c++                arm-linux-gcjh         arm-linux-objdump
arm-linux-c++filt            arm-linux-gcov         arm-linux-ranlib
arm-linux-cpp                arm-linux-gnatbind     arm-linux-readelf
arm-linux-elf2flt            arm-linux-grepjar      arm-linux-size
arm-linux-flthdr             arm-linux-jar          arm-linux-strings
arm-linux-g++                arm-linux-jcf-dump     arm-linux-strip
```

1、准备交叉编译环境 III

- 查看版本信息

```
./usr/local/bin/arm-linux-gcc -v
```

```
Reading specs from
/home/xlanchen/workspace/usr/local/bin/../lib/gcc/arm-linux/3.4.4/specs
Configured with: ../configure —target=arm-linux —disable-shared —prefix=/usr/local
—with-headers=/home/gerg/new-wave.ixdp425/linux-2.4.x/include —with-gnu-as
—with-gnu-ld —enable-multilib
Thread model: posix
gcc version 3.4.4
```

编译arm1inux

- 从kerne1.org上找到china的镜像网站，下载1inux-2.6.26.tar.bz2
- 在工作目录$WDIR中解压缩，后执行如下命令
(或者使用CROSS_COMPILE=arm-1inux-gnueabi-)

```
cd 1inux-2.6.26
make ARCH=arm CROSS_COMPILE=arm-1inux- s3c2410_defconfig
make ARCH=arm CROSS_COMPILE=arm-1inux- menuconfig
```

编译armlinux

- 修改如下:

  - 修改 include/asm-arm/arch-s3c2410/map.h
    #define S3C2410_CS6 (0xC0000000UL)
  - 修改 include/asm-arm/arch-s3c2410/memory.h
    #define PHYS_OFFSET (0xC0000000UL)
  - Boot options→Default kernel command string:
    mem=32M console=ttySAC0 root=/dev/ram initrd=0xc0800000,0x00800000 rw
  - Device Driver→Character Driver→Serial Driver，取消8250/16550 and compatible serial support选项
  - [可选]File systems→中，仅仅保留ext2
  - [可选]Networking→，取消Networking support
  - [可选]Device Driver→，取消I2C support和USB support

编译arm1inux

- 编译

  make ARCH=arm CROSS_COMPILE=arm-1inux-

  需要等待较长时间

准备根文件系统

- 在后面的课程中准备

Outline

1、准备交叉编译环境 I

- 下载arm-linux-tools-20061213.tar.gz
- 在主机的系统根目录下或在当前目录下解压缩

```
sudo tar zvxf arm-linux-tools-20061213.tar.gz
```

- 查看安装是否成功

./usr/local/bin/arm-linux-<tab>

```
arm-linux-addr2line          arm-linux-g77          arm-linux-jv-scan
arm-linux-addr2name.awk      arm-linux-gcc          arm-linux-ld
arm-linux-ar                 arm-linux-gcc-3.4.4    arm-linux-ld.real
arm-linux-arm-linux-gcjh     arm-linux-gccbug       arm-linux-nm
arm-linux-as                 arm-linux-gcj          arm-linux-objcopy
arm-linux-c++                arm-linux-gcjh         arm-linux-objdump
arm-linux-c++filt            arm-linux-gcov         arm-linux-ranlib
arm-linux-cpp                arm-linux-gnatbind     arm-linux-readelf
arm-linux-elf2flt            arm-linux-grepjar      arm-linux-size
arm-linux-flthdr             arm-linux-jar          arm-linux-strings
arm-linux-g++                arm-linux-jcf-dump     arm-linux-strip
```

1、准备交叉编译环境 II

- 查看版本信息

```
./usr/local/bin/arm-linux-gcc -v
```

```
Reading specs from
/home/xlanchen/workspace/usr/local/bin/../lib/gcc/arm-linux/3.4.4/specs
Configured with: ../configure —target=arm-linux —disable-shared —prefix=/usr/local
—with-headers=/home/gerg/new-wave.ixdp425/linux-2.4.x/include —with-gnu-as
—with-gnu-ld —enable-multilib
Thread model: posix
gcc version 3.4.4
```

2、配置并编译uClinux

1. 下载源代码uClinux-dist-20140504.tar.bz2
2. 解压缩tar -jvxf uClinux-dist-20140504.tar.bz2
3. 配置并编译

```
export LDLIBS=-ldl（可能需要）
make xconfig
```

在vendor/product选项中选择GDB/ARMulator
Kernel版本选择2.4.x
其他选项不变（使用缺省选项）

```
make dep; make
```

2、配置并编译uClinux

4. 查看编译出来的内核映像
   - images目录下

```
images/
├──── boot.rom
├──── linux
└──── romfs-inst.log
```

   - linux-2.4.x目录下的linux

看一看Linux的链接命令，了解linux文件的组成

```
arm-linux-ld -p -X -T arch/armnommu/vmlinux.lds arch/armnommu/kernel/head-armv.o
arch/armnommu/kernel/init_task.o init/main.o init/version.o init/do_mounts.o \
—start-group \ arch/armnommu/kernel/kernel.o arch/armnommu/mm/mm.o
arch/armnommu/mach-atmel/atmel.o kernel/kernel.o mmnommu/mmnommu.o fs/fs.o ipc/ipc.o \
drivers/char/char.o drivers/serial/serial.o drivers/block/block.o drivers/misc/misc.o
drivers/net/net.o drivers/media/media.o \ net/network.o \ arch/armnommu/lib/lib.a
/media/xlanchen/store/work/5教学/2014Fall1EmbeddedOS/uClinux-dist/linux-2.4.x/lib/lib.a
/usr/local/lib/gcc/arm-linux/3.4.4/soft-float/libgcc.a \ —end-group \ -o linux
```

看一看romfs的生成命令，了解romfs来源

```
genromfs -v -V ”ROMdisk” -f
/media/xlanchen/store/work/5教学/2014Fall1EmbeddedOS/uClinux-dist/images/boot.rom -d
/media/xlanchen/store/work/5教学/2014Fall1EmbeddedOS/uClinux-dist/romfs
```

看一看romfs中的内容

1. 到romfs目录看看
2. 根据编译的输出信息查看
3. 挂载起来看一看

编译输出信息如下：

<目录深度><文件/目录名>[设备号，inode节点号]<mode><size>[link]

```
0 ROMdisk [0xffffffff, 0xffffffff] 37777777777, sz 0, at 0x0
1 . [0x824 , 0x8e3c42 ] 0040775, sz 0, at 0x20
1 .. [0x824 , 0x8e2a1a ] 0040755, sz 0, at 0x40 [link to 0x20 ]
1 etc [0x824 , 0x8e3c45 ] 0040775, sz 0, at 0x60
2 rc [0x824 , 0x8e3c86 ] 0100644, sz 207, at 0x80
2 .. [0x824 , 0x8e3c42 ] 0040775, sz 0, at 0x170 [link to 0x20 ]
2 passwd [0x824 , 0x8e3c8a ] 0100644, sz 21, at 0x190
2 . [0x824 , 0x8e3c45 ] 0040775, sz 0, at 0x1d0 [link to 0x60 ]
2 version [0x824 , 0x8e3c8b ] 0100664, sz 73, at 0x1f0
2 inetd.conf [0x824 , 0x8e3ca3 ] 0100664, sz 44, at 0x260
2 ramfs.img [0x824 , 0x8e3c9f ] 0100644, sz 2874, at 0x2b0
2 services [0x824 , 0x8e3c88 ] 0100644, sz 8205, at 0xe10
2 motd [0x824 , 0x8e3c89 ] 0100644, sz 343, at 0x2e40
2 inittab [0x824 , 0x8e3c87 ] 0100644, sz 49, at 0x2fc0
1 bin [0x824 , 0x8e3c43 ] 0040775, sz 0, at 0x3020
2 telnetd [0x824 , 0x8e3ca2 ] 0100744, sz 45764, at 0x3040
2 [0x824 , 0x8e3c42 ] 0040775, sz 0, at 0xe330 [link to 0x20 ]
```

3、在skyeye上运行uClinux

- 从skyeye-testsuite-1.2.5/uClinux/at91/uclinux_cs8900a/目录下复制skyeye.conf和romfs.img
- 运行：

```
skyeye -e linux -c skyeye.conf
```

- 直接使用编译生成的boot.rom失败
  但由于配置不一致，在进入shell之前有个别错误。

3、在skyeye上运行uClinux

- skyeye.conf的内容如下：

skyeye.conf

```
#skyeye config file sample
cpu: arm7tdmi

mach: at91

mem_bank: map=M, type=RW, addr=0x00000000, size=0x00004000
mem_bank: map=M, type=RW, addr=0x01000000, size=0x00400000
mem_bank: map=M, type=R, addr=0x01400000, size=0x00400000, file=./romfs.img
mem_bank: map=M, type=RW, addr=0x02000000, size=0x00400000
mem_bank: map=M, type=RW, addr=0x02400000, size=0x00008000
mem_bank: map=M, type=RW, addr=0x04000000, size=0x00400000
mem_bank: map=I, type=RW, addr=0xf0000000, size=0x10000000
#set nic info
#net: type=cs8900a, base=0xfffa0000, size=0x20,int=16, mac=0:4:3:2:1:f,
ethmod=tuntap, hostip=10.0.0.1
net: type=cs8900a, ethmod=tuntap, hostip=10.0.0.1
#dbct: state=on
```

3、在skyeye上运行uClinux

- 进入uClinux界面如下：

```
Welcome to

        ____ _  _
       /  __| ||_|
   _   _| |  | | _ ____  _   _  _  _
  | | | | |  | || |  _ \| | | |\ \/ /
  | |_| | |__| || | | | | |_| |/    \
  |  ___\____|_||_|_| |_|\____|\_/\_/
  | |
  |_|

GDB/ARMulator support by <davidm@snapgear.com>
For further information check:
http://www.uclinux.org/

Command: /bin/ifconfig eth0 up 10.0.0.2
SIOCGIFFLAGS: No such device
SIOCSIFADDR: No such device
pid 11: failed 512
Execution Finished, Exiting
init: Booting to single user mode

Sash command shell (version 1.1.1)
/>
```

Outline

小结

1. SkyEye简介
2. SkyEye的安装
   - SkyEye的安装
   - 试用skyeye
   - 下载skyeye源代码，编译并安装
3. 启动Linux执行自己的程序
   - 添加hello到uclinux的根文件系统中
   - 在ArmLinux中运行hello
4. 编译linux
   - 编译armlinux
   - 编译μCLinux
5. 小结和作业

Project3

- 在skyeye上成功跑出arm1inux和μCLinux
  - 可以使用现成的映像
  - 〔可选，加分〕可以自己编译
- 分别将hello加入到arm1inux和μCLinux的根文件系统中，在skyeye上
- 提交报告，要求要说明
  - 被使用的arm1inux和μCLinux的版本和Linux内核的版本
  - 〔可选，加分〕若自己编译了内核，给出编译arm1inux和μCLinux的交
  - 编译hello的交叉编译器的版本和编译过程
  - 给出关键输出的图示
- 难度：

  arm1inux>μCLinux>使用现成的内核

Thanks !

The end.