# 实验一USB转串口驱动的安装

**【实验目的】**

在Windows中安装USB转串口的驱动程序。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

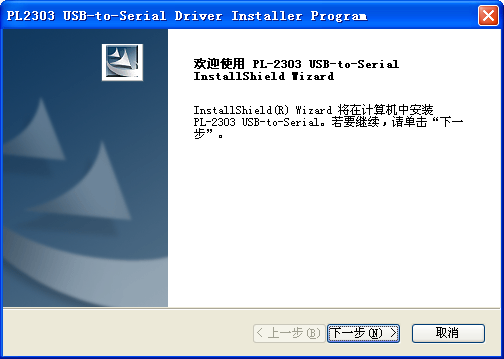
1. Windows
2. FS4412平台

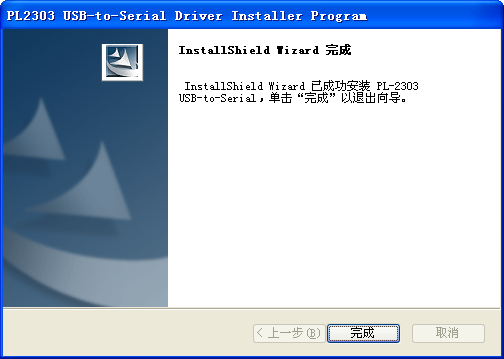
**【实验步骤】**

1. 双击“案例源码\第一天\驱动程序\CH341SER.EXE”，弹出如下图所示的窗口，点击安装。即可安装CH341的USB转串口驱动程序，如下面的图所示。



1. 双击“案例源码\第一天\驱动程序\ PL2303\_Prolific\_DriverInstaller\_v1.12.0.exe”，弹出如下图所示的窗口，点击下一步。即可安装PL2303的USB转串口驱动程序，如下面的图所示。





1. 将USB转串口线插入到PC机的USB接口，在设备管理器中会看到相应的串口端口号。如果是CH341的串口线，则串口的端口如下图所示。



如果是PL2303的串口线，则串口的端口如下图所示。



具体的串口号根据系统和接入的USB接口而定。

# 实验二常用串口终端工具安装

**【实验目的】**

在Windows中安装串口终端工具，掌握常见串口终端工具的配置和使用方法。

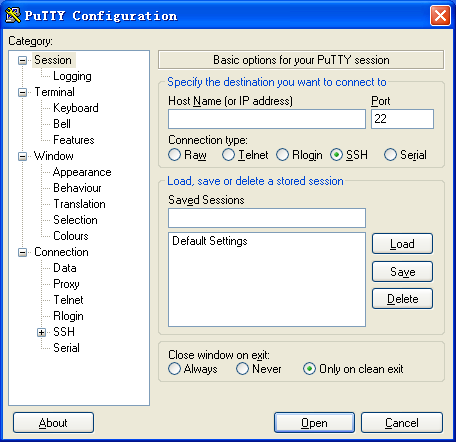
说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

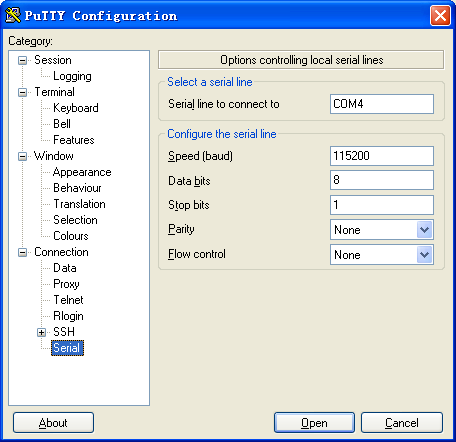
1. Windows
2. Ubuntu 12.04发行版
3. FS4412平台

**【实验步骤】**

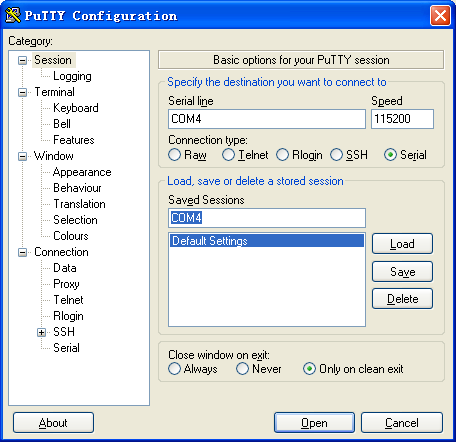
1. 将“案例源码\第一天\工具软件\putty.exe”拷贝到桌面，或拷贝至任一文件夹然后发送快捷方式到桌面。
2. 双击桌面上的putty.exe图标，弹出如下的对话框。



1. 点击左侧的Serial列表选项，按下图进行设置，注意串口号按照设备管理器中查看的而定。



1. 点击左侧的Session列表选项，按下图进行设置。注意“Saved Sessions”文本框中的名字是该会话的名字，可随意指定。然后点击Save按钮保存该会话设置。最后点Open按钮，打开该会话。



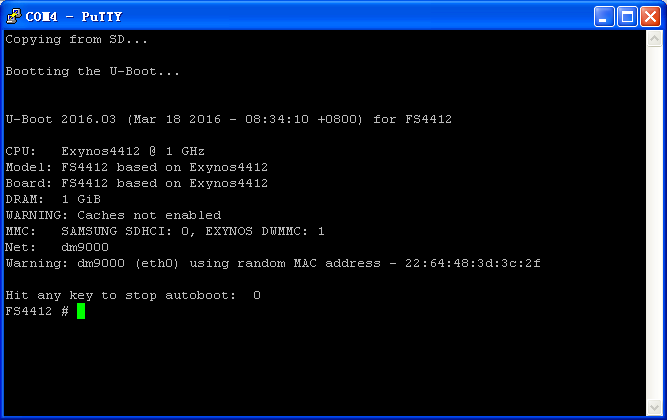
1. 将USB转串口线接至FS4412开发板的CON7 DB9串口插座上，设置好启动方式。按照如下拨码为eMMC启动。



按照如下拨码，则为SD卡启动。

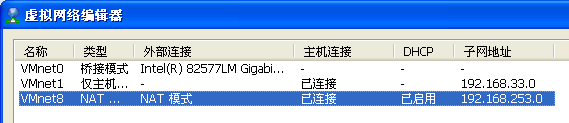


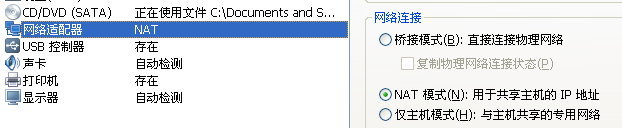
1. 将开发板接上电源，在putty中可以看到如下的打印信息，说明串口驱动和串口终端软件安装并配置成功。



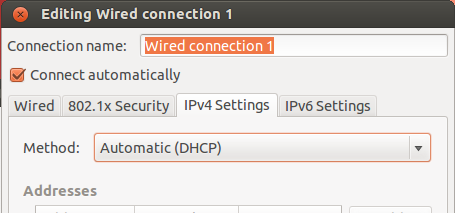
1. 如果是在纯Linux环境下开发，则可以安装minicom串口终端软件。安装前首先确保Ubuntu能够连接上外网。如果是虚拟机，则可以做以下设置。







上面是将虚拟机中的网卡设为NAT连接方式。

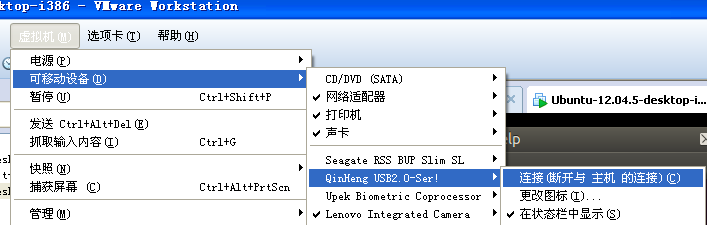


上图是设置虚拟机中的网卡自动获得IP地址。

1. 网络设置成功后，使用下面的命令安装minicom串口终端软件。

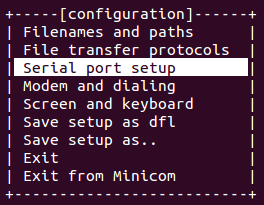
$ sudo apt-get install minicom

1. 如果是Ubuntu虚拟机，则通过如下的方式将USB转串口线连接至虚拟机。



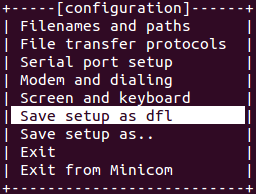
1. 连接成功后运行minicom程序，进行配置。

$ sudo minicom -s

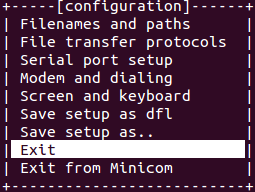




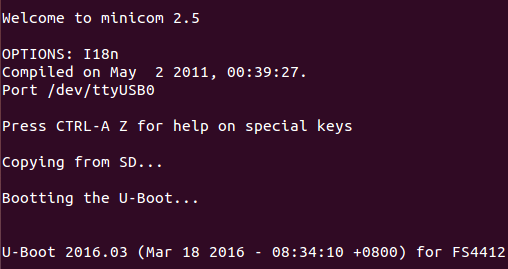
上面的修改方法是：按下对应的字母键，然后进行编辑，按回车完成编辑。再次按回车，退出配置菜单界面。



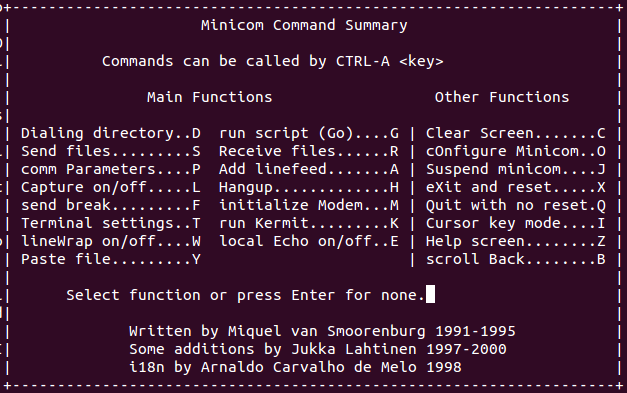
在主界面中选择保存配置，然后回车。



开发板上电，可以看到如下打印信息。



按下CTRL-A后，再Z键，可以出现帮助菜单。



按W键，打开自动换行。按Q键退出minicom程序。

1. 之后只需要执行下面的命令即可以运行minicom串口终端程序。

$ sudo minicom

# 实验三 TFTP服务器的安装和测试

**【实验目的】**

掌握在Ubuntu中安装TFTP服务器的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 运行下面的命令安装TFTP服务器和客户端。

$ sudo apt-get install tftpd-hpa tftp-hpa

1. 运行下面的命令，重启TFTP服务器。

$ sudo service tftpd-hpa restart

1. 运行下面的命令，新建一个文件，并将其移动到TFTP服务器的默认上传下载目录。

$ echo "tftp test" > test.txt

$ sudo mv test.txt /var/lib/tftpboot/

1. 运行下面的命令，从服务器上下载test.txt文件，并退出tftp程序。

$ tftp localhost

tftp> get test.txt

tftp> q

1. 运行下面的命令，确认下载的文件内容正确。

$ cat test.txt

tftp test

1. 如果TFTP的下载不成功，运行下面的命令卸载软件（连同配置信息一起），然后再重新安装，最后再重启TFTP服务器。

$ sudo apt-get remove --purge tftpd-hpa tftp-hpa

$ sudo apt-get install tftpd-hpa tftp-hpa

$ sudo service tftpd-hpa restart

# 实验四 NFS服务器的安装和测试

**【实验目的】**

掌握在Ubuntu中安装NFS服务器的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 运行下面的命令安装NFS服务器。

$ sudo apt-get install nfs-kernel-server

1. 运行下面的命令，创建一个目录，并在该文件下创建一个文件。

$ sudo mkdir /nfs

$ sudo chown farsight /nfs # 非root用户需要执行该步骤，farsight为用户名

$ sudo chgrp farsight /nfs # 非root用户需要执行该步骤，farsight为用户组

$ mkdir /nfs/rootfs

$ echo "nfs test" > /nfs/rootfs/test.txt

1. 编辑/etc/exports配置文件。

$ sudo vim /etc/exports

添加如下内容：

/nfs/rootfs \*(rw,sync,no\_subtree\_check,no\_root\_squash)

其中：

/nfs/rootfs：共享的目录；

\*：不限定客户端；

rw：共享目录可读可写；

sync：将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性；

no\_subtree\_check ：即使输出目录是一个子目录，nfs服务器也不检查其父目录的权限，这样可以提高效率；

no\_root\_squash：来访的root用户保持root帐号权限；

1. 使用下面的命令，重启NFS服务。

$ sudo service nfs-kernel-server restart

1. 使用下面的命令，将共享目录挂在到/mnt目录下，并修文件。

$ sudo mount -t nfs localhost:/nfs/rootfs /mnt

$ vim /mnt/test.txt

1. 使用下面的命令，查看原来的文件已经被修改。

$ cat /nfs/rootfs/test.txt

1. 使用下面的命令取消挂载。

$ sudo umount /mnt

# 实验五安装ARM交叉编译工具链和其他工具

**【实验目的】**

掌握ARM交叉编译工具链和其他工具的安装方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将交叉编译工具链压缩包拷贝至Ubuntu主机的用户主目录下（或其他任意目录），然后进入该目录使用下面的命令对该压缩包进行解压。

$ tar –xvf gcc-4.6.4.tar.xz

解压完成后，进入到相应的目录获取绝对路径。

$ cd gcc-4.6.4/bin/

$ pwd

/home/farsight/gcc-4.6.4/bin

编辑/etc/environment文件，加入相应的路径。

$ sudo vim /etc/environment

PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/farsight/gcc-4.6.4/bin"

注销后重新登录，使用下面的命令确定交叉编译工具链安装成功。

$ arm-none-linux-gnueabi-gcc -v

Using built-in specs.

COLLECT\_GCC=arm-none-linux-gnueabi-gcc

COLLECT\_LTO\_WRAPPER=/home/farsight/gcc-4.6.4/bin/../libexec/gcc/arm-arm1176jzfssf-linux-gnueabi/4.6.4/lto-wrapper

Target: arm-arm1176jzfssf-linux-gnueabi

Configured with: /work/builddir/src/gcc-4.6.4/configure --build=i686-build\_pc-linux-gnu --host=i686-build\_pc-linux-gnu --target=arm-arm1176jzfssf-linux-gnueabi --prefix=/opt/TuxamitoSoftToolchains/arm-arm1176jzfssf-linux-gnueabi/gcc-4.6.4 --with-sysroot=/opt/TuxamitoSoftToolchains/arm-arm1176jzfssf-linux-gnueabi/gcc-4.6.4/arm-arm1176jzfssf-linux-gnueabi/sysroot --enable-languages=c,c++ --with-arch=armv6zk --with-cpu=arm1176jzf-s --with-tune=arm1176jzf-s --with-fpu=vfp --with-float=softfp --with-pkgversion='crosstool-NG hg+default-2685dfa9de14 - tc0002' --disable-sjlj-exceptions --enable-\_\_cxa\_atexit --disable-libmudflap --disable-libgomp --disable-libssp --disable-libquadmath --disable-libquadmath-support --with-gmp=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-mpfr=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-mpc=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-ppl=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-cloog=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-libelf=/work/builddir/arm-arm1176jzfssf-linux-gnueabi/buildtools --with-host-libstdcxx='-static-libgcc -Wl,-Bstatic,-lstdc++,-Bdynamic -lm' --enable-threads=posix --enable-target-optspace --without-long-double-128 --disable-nls --disable-multilib --with-local-prefix=/opt/TuxamitoSoftToolchains/arm-arm1176jzfssf-linux-gnueabi/gcc-4.6.4/arm-arm1176jzfssf-linux-gnueabi/sysroot --enable-c99 --enable-long-long

Thread model: posix

gcc version 4.6.4 (crosstool-NG hg+default-2685dfa9de14 - tc0002)

1. 安装curses开发库，在后面进行源码配置时需要用到该库，使用下面的命令。

$ sudo apt-get install libncurses5-dev

1. 安装dtc工具，用于编译U-Boot的设备树，使用下面的命令。

首先获取自动化安装脚本文件，如果不能下载，请直接使用提供的文件（位于“案例源码\第一天\程序源码”）。

$ wget -c <https://raw.github.com/RobertCNelson/tools/master/pkgs/dtc.sh>

添加可执行权限。

$ chmod u+x dtc.sh

运行自动化安装脚本进行安装。

$ ./dtc.sh

运行下面的命令，确认安装成功。

$ dtc –v

Version: DTC 1.4.0-gf345d9e4

1. 安装mkimage工具，用于生成uImage。将mkimage（位于案例源码\第一天\镜像文件\mkimage）拷贝到Ubuntu主机中的/usr/local/bin目录即可。

# 实验六制作启动SD卡

**【实验目的】**

掌握启动SD卡的制作方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将“案例源码\第一天\程序源码\sdtool”整个文件夹拷贝到Ubuntu虚拟机中的一个目录中。
2. 在Ubuntu中打开一个终端，并进入到sdtool目录。

$ cd sdtool/

1. 将SD卡插入USB读卡器中，将读卡器插入PC机，使用前面的方法将SD卡连接到虚拟机中。
2. 在终端中执行下面的命令烧写SD卡。

$ sudo ./sdtool.sh fuse /dev/sdb u-boot-fs4412.bin

---------------------------------------

U-Boot fusing

766+1 records in

766+1 records out

392244 bytes (392 kB) copied, 4.70365 s, 83.4 kB/s

---------------------------------------

Image is fused successfully.

Eject SD card and insert it again.

上面的/dev/sdb是SD卡设备，要根据自己的实际情况决定，最好不要连接U盘等其他USB外设，否则可能会将镜像烧写在U盘上！！！

1. 将制作好的启动SD卡插入FS4412开发板的SD卡插槽中，选择从SD卡启动。然后开发板上电，查看串口终端上的信息，确认启动SD卡制作成功。

# 实验七常用U-Boot命令练习

**【实验目的】**

掌握常用的U-Boot命令。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 使用上一实验制作的启动SD卡启动开发板。
2. 输入？或help查看命令帮助。

FS4412 #?

? - alias for 'help'

base - print or set address offset

bdinfo - print Board Info structure

boot - boot default, i.e., run 'bootcmd'

bootd - boot default, i.e., run 'bootcmd'

……

ums - Use the UMS [USB Mass Storage]

version - print monitor, compiler and linker version

1. 使用printenv命令查看环境变量。

FS4412 # printenv

baudrate=115200

bootcmd=tftpboot 40000000 secure.bin; tftpboot 40008000 nonsecure.bin; go 4000000

bootdelay=2

ethact=dm9000

fdtcontroladdr=7feb00b0

gatewayip=192.168.10.1

ipaddr=192.168.0.150

netmask=255.255.255.0

serverip=192.168.0.103

Environment size: 257/16380 bytes

1. 使用setenv命令修改环境变量、新增环境变量、删除环境变量。

FS4412 # setenv serverip 192.168.10.100

FS4412 # setenv ipaddr 192.168.10.200

FS4412 # setenv myenv envtest

FS4412 # printenv

……

ipaddr=192.168.10.200

myenv=envtest

serverip=192.168.10.100

……

FS4412 # setenv myenv

FS4412 # printenv

……

ipaddr=192.168.10.200

serverip=192.168.10.100

……

1. 使用saveenv保存环境变量，保存成功后重启开发板，查看环境变量信息。

FS4412 # saveenv

Saving Environment to MMC...

Writing to MMC(0)... done

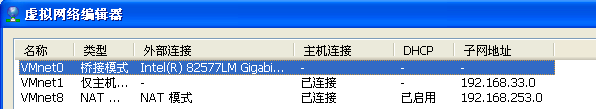
……

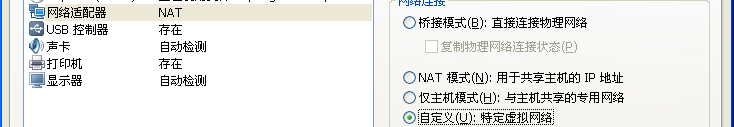
FS4412 # printenv

……

1. 使用ping命令，判断和Ubuntu主机的网络连接是否正常。

Ubuntu主机的网络可以按如下方式进行设置。





以上步骤设置Ubuntu虚拟机的网卡是以桥接方式进行连接的。



以上是手动设置Ubuntu虚拟机的网络IP地址。

FS4412 # setenv serverip 192.168.10.100

FS4412 # setenv ipaddr 192.168.10.200

FS4412 # saveenv

Saving Environment to MMC...

Writing to MMC(0)... done

FS4412 # ping 192.168.10.100

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 26:35:8b:7b:12:b3

operating at 100M full duplex mode

Using dm9000 device

host 192.168.10.100 is alive

1. 使用tftpboot命令，从Ubuntu主机上下载文件到开发板的内存中。

按照实验三的步骤安装并配置好TFTP服务器，按照前面的实验设置好环境变量并确保能ping通Ubuntu主机。

在Ubuntu主机上将要下载的文件拷贝到TFTP服务器指定的下载目录，如将Linux内核镜像文件uImage、设备树文件exynos4412-fs4412.dtb和根文件系统的镜像文件ramdisk.img）（这些文件位于“案例源码\第一天\镜像文件”）拷贝至Ubuntu主机的/var/lib/tftpboot目录，使用如下命令。

$ sudo cp uImage /var/lib/tftpboot/

$ sudo cp exynos4412-fs4412.dtb /var/lib/tftpboot/

$ sudo cp ramdisk.img /var/lib/tftpboot/

在串口终端上运行下面的命令下载文件。

FS4412 # tftpboot 41000000 uImage

FS4412 # tftpboot 42000000 exynos4412-fs4412.dtb

FS4412 # tftpboot 43000000 ramdisk.img

成功下载会在串口终端上看到如下输出。

FS4412 # tftpboot 41000000 uImage

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 26:35:8b:7b:12:b3

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'uImage'.

Load address: 0x41000000

Loading: #################################################################

#################################################################

#################################################################

############################

779.3 KiB/s

done

Bytes transferred = 3261232 (31c330 hex)

FS4412 # tftpboot 42000000 exynos4412-fs4412.dtb

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 26:35:8b:7b:12:b3

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'exynos4412-fs4412.dtb'.

Load address: 0x42000000

Loading: ###

316.4 KiB/s

done

Bytes transferred = 34402 (8662 hex)

FS4412 # tftpboot 43000000 ramdisk.img

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 26:35:8b:7b:12:b3

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'ramdisk.img'.

Load address: 0x43000000

Loading: #################################################################

#################################################################

############################################

802.7 KiB/s

done

Bytes transferred = 2544147 (26d213 hex)

1. 使用bootm命令启动内核。在上一步骤的基础上，在串口终端中使用下面的命令来启动Linux系统。

FS4412 # bootm 41000000 43000000 42000000

## Booting kernel from Legacy Image at 41000000 ...

Image Name: Linux-3.14.25

……

[ 2.180000] hub 1-3:1.0: 3 ports detected

[root@farsight ]#

1. 使用go命令来运行bin文件。将u-boot.bin文件拷贝到Ubuntu主机的/var/lib/tftpboot目录，使用如下命令。

$ sudo cp u-boot.bin /var/lib/tftpboot/

开发板启动后，在串口终端中执行下面的命令下载u-boot.bin文件到开发板的内存中，并运行。

FS4412 # tftpboot 43E00000 u-boot.bin

FS4412 # go 43E00000

成功执行后，串口终端看到如下输出。

FS4412 # tftpboot 43E00000 u-boot.bin

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 26:35:8b:7b:12:b3

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'u-boot.bin'.

Load address: 0x43e00000

Loading: ##########################

717.8 KiB/s

done

Bytes transferred = 369292 (5a28c hex)

FS4412 # go 43E00000

## Starting application at 0x43E00000 ...

U-Boot 2016.03 (Apr 29 2016 - 14:57:26 +0800) for FS4412

CPU: Exynos4412 @ 1 GHz

Model: FS4412 based on Exynos4412

Board: FS4412 based on Exynos4412

DRAM: 1 GiB

WARNING: Caches not enabled

MMC: EXYNOS DWMMC: 0, SAMSUNG SDHCI: 1

Net: dm9000

Warning: dm9000 (eth0) using random MAC address - e2:33:0d:71:3c:f7

Hit any key to stop autoboot: 0

1. 使用mmc命令烧写eMMC。

将最终烧写的U-Boot镜像文件拷贝至Ubuntu主机的/var/lib/tftpboot目录，使用如下命令。

$ sudo cp u-boot-fs4412.bin /var/lib/tftpboot/

从SD卡启动，在串口终端上执行下面的命令将文件下载到开发板的内存中。

FS4412 # tftpboot 41000000 u-boot-fs4412.bin

使用下面的命令查看mmc设备。

FS4412 # mmc list

SAMSUNG SDHCI: 0 (SD)

EXYNOS DWMMC: 1

上面表示设备0是SD卡，设备1是eMMC。这是从SD卡启动U-Boot的情况。

使用下面的命令，切换当前操作的设备，当前要操作的是eMMC，使用下面的命令进行切换。

FS4412 # mmc dev 1

switch to partitions #0, OK

mmc1 is current device

使用下面的命令开启eMMC的boot分区。

FS4412 # mmc partconf 1 1 1 1

FS4412 # mmc bootbus 1 1 0 0

使用下面的命令将U-Boot的镜像烧写至eMMC。

FS4412 # mmc write 41000000 0 400

使用下面的命令关闭boot分区。

FS4412 # mmc partconf 1 1 1 0

（注意：如果本身就是从eMMC启动的，则mmc partconf、mmc bootbus命令的第一个参数1要改为0。mmc bootbus命令需要慎用，如果写错了将会导致eMMC不能启动，需要从SD卡启动u-boot后将eMMC重新开启。）

开发板掉电，将开关拨至eMMC启动，重新启动开发板。串口终端看到如下启动信息。

Copying from eMMC...

Bootting the U-Boot...

U-Boot 2016.03 (Apr 29 2016 - 14:57:26 +0800) for FS4412

CPU: Exynos4412 @ 1 GHz

Model: FS4412 based on Exynos4412

Board: FS4412 based on Exynos4412

DRAM: 1 GiB

WARNING: Caches not enabled

MMC: EXYNOS DWMMC: 0, SAMSUNG SDHCI: 1

Net: dm9000

Warning: dm9000 (eth0) using random MAC address - e2:33:0d:71:3c:f7

Hit any key to stop autoboot: 0

# 实验八通过NFS挂载根文件系统

**【实验目的】**

掌握bootcmd、bootargs环境变量的设置和NFS挂载根文件系统的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将“案例源码\第一天\镜像文件\rootfs.tar.xz”拷贝至Ubuntu主机的/nfs目录中。
2. 如果/nfs目录下有rootfs目录，使用下面的命令删除。

$ rm -rf /nfs/rootfs

1. 进入到/nfs目录，解压根文件系统的压缩文件。

$ cd /nfs/

$ tar -xvf rootfs.tar.xz

1. 按照实验四的方法将/nfs/rootfs目录设置为NFS共享目录。
2. 开发板上电后，运行U-Boot，设置bootcmd环境变量和bootargs环境变量。

FS4412 # setenv bootcmd tftpboot 41000000 uImage\; tftpboot 42000000 exynos4412-fs4412.dtb\; bootm 41000000 - 42000000

FS4412 # setenv bootargs noinitrd root=/dev/nfs nfsroot=192.168.10.100:/nfs/rootfs rw console=ttySAC2,115200 init=/linuxrc ip=192.168.10.200

FS4412 # saveenv

1. 在串口终端中使用下面的命令来自动下载文件并挂载根文件系统。

FS4412 # boot

1. 如果重启开发板，不需要任何操作则可以自动启动Linux系统。

# 实验九 U-Boot的配置和编译

**【实验目的】**

掌握U-Boot的配置和编译的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将“案例源码\第二天\程序源码\u-boot-2016.03.tar.bz2”拷贝至Ubuntu主机中。
2. 进入到相应的目录，使用下面的命令对源码的压缩包进行解压。

$ tar -jxvf u-boot-2016.03.tar.bz2

1. 进入到解压的目录，使用下面的命令生成默认的配置。

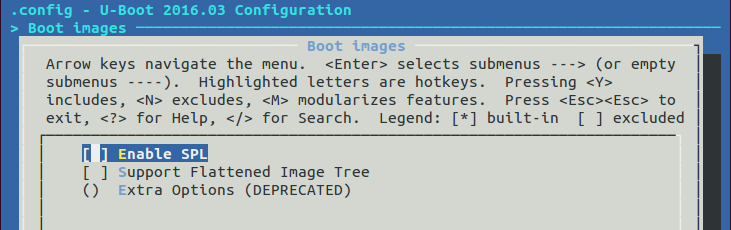
$ cd u-boot-2016.03

$ make origen\_defconfig

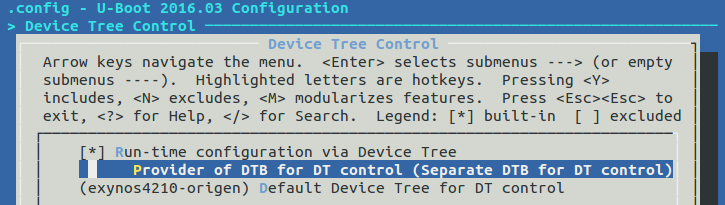
1. 使用下面的命令生成U-Boot的配置界面，并对配置进行修改。

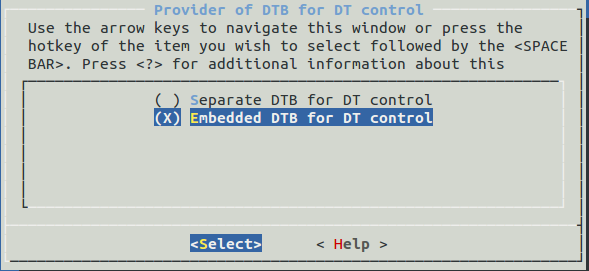
$ make menuconfig

取消SPL选项。

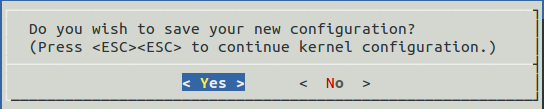


将设备树合并到U-Boot镜像中。





保存并退出配置。



1. 使用下面的命令编译U-Boot。

$ make CROSS\_COMPILE=arm-none-linux-gnueabi-

如果不想每次都在命令行中设置交叉编译工具，可以编辑U-Boot源码树顶层目录下的Makefile文件，在：

ifeq ($(HOSTARCH),$(ARCH))

CROSS\_COMPILE ?=

endif

后添加：

ARCH = arm

ifeq (arm,$(ARCH))

CROSS\_COMPILE ?= arm-none-linux-gnueabi-

endif

1. 将编译好的u-boot.bin文件拷贝到TFTP服务器指定的下载目录。

$ sudo cp u-boot.bin /var/lib/tftpboot/

1. 开发板上电，在串口终端中执行下面的命令下载并测试新编译的U-Boot。

FS4412 # tftpboot 43E00000 u-boot.bin

FS4412 # go 43E00000

运行的效果如下。

FS4412 # tftpboot 43E00000 u-boot.bin

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: c2:32:c8:3e:ed:27

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'u-boot.bin'.

Load address: 0x43e00000

Loading: ########################

738.3 KiB/s

done

Bytes transferred = 339756 (52f2c hex)

FS4412 # go 43E00000

## Starting application at 0x43E00000 ...

U-Boot 2016.03 (Apr 29 2016 - 20:52:50 +0800) for ORIGEN

CPU: Exynos4412 @ 1 GHz

Model: Insignal Origen evaluation board based on Exynos4210

Board: Insignal Origen evaluation board based on Exynos4210

DRAM: 1 GiB

WARNING: Caches not enabled

MMC: DWMMC56: Can't get bus-width

exynos\_dwmci\_process\_node: failed to decode dev 0

SAMSUNG SDHCI: 0

mmc fail to send stop cmd

\*\*\* Warning - read failed, using default environment

Hit any key to stop autoboot: 0

ORIGEN #

# 实验十 U-Boot基础移植

**【实验目的】**

熟悉U-Boot的基本移植过程。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将“案例源码\第二天\程序源码\u-boot-2016.03.tar.bz2”拷贝至Ubuntu主机中。
2. 进入到相应的目录，使用下面的命令对源码的压缩包进行解压。

$ tar -jxvf u-boot-2016.03.tar.bz2

1. 进入到解压的目录，拷贝相应的文件。

$ cp configs/origen\_defconfig configs/fs4412\_defconfig

$ cp arch/arm/dts/exynos4210-origen.dts arch/arm/dts/exynos4412-fs4412.dts

$ cp -a board/samsung/origen/ board/samsung/fs4412

$ mv board/samsung/fs4412/origen.c board/samsung/fs4412/fs4412.c

$ rm -rf board/samsung/fs4412/tools/

$ cp include/configs/origen.h include/configs/fs4412.h

1. 修改相应的文件。
   1. 修改arch/arm/mach-exynos/Kconfig

$ vim arch/arm/mach-exynos/Kconfig

在：

config TARGET\_ORIGEN

bool "Exynos4412 Origen board"

select SUPPORT\_SPL

后面添加：

config TARGET\_FS4412

bool "Exynos4412 FS4412 board"

在：

source "board/samsung/origen/Kconfig"

后面添加：

source "board/samsung/fs4412/Kconfig"

* 1. 修改configs/fs4412\_defconfig

将：

CONFIG\_TARGET\_ORIGEN=y

CONFIG\_DEFAULT\_DEVICE\_TREE="exynos4210-origen"

CONFIG\_SPL=y

CONFIG\_SYS\_PROMPT="ORIGEN # "

改为：

CONFIG\_TARGET\_FS4412=y

CONFIG\_DEFAULT\_DEVICE\_TREE="exynos4412-fs4412"

CONFIG\_SYS\_PROMPT="FS4412 # "

* 1. 修改arch/arm/dts/exynos4412-fs4412.dts

将：

#include "exynos4210.dtsi"

/ {

model = "Insignal Origen evaluation board based on Exynos4210";

compatible = "insignal,origen", "samsung,exynos4210";

chosen {

bootargs ="";

};

改为：

#include "exynos4412.dtsi"

/ {

model = "FS4412 based on Exynos4412";

compatible = "samsung,fs4412", "samsung,exynos4412";

* 1. 修改arch/arm/dts/Makefile

在：

exynos4412-trats2.dtb \

后添加：

exynos4412-fs4412.dtb \

* 1. 修改board/samsung/fs4412/Kconfig

将：

if TARGET\_ORIGEN

config SYS\_BOARD

default "origen"

config SYS\_VENDOR

default "samsung"

config SYS\_CONFIG\_NAME

default "origen"

endif

改为：

if TARGET\_FS4412

config SYS\_BOARD

default "fs4412"

config SYS\_VENDOR

default "samsung"

config SYS\_CONFIG\_NAME

default "fs4412"

endif

* 1. 修改board/samsung/fs4412/Makefile

将：

ifdef CONFIG\_SPL\_BUILD

# necessary to create built-in.o

obj- := \_\_dummy\_\_.o

hostprogs-y := tools/mkorigenspl

always := $(hostprogs-y)

# omit -O2 option to suppress

# warning: dereferencing type-punned pointer will break strict-aliasing rules

#

# TODO:

# Fix the root cause in tools/mkorigenspl.c and delete the following work-around

$(obj)/tools/mkorigenspl: HOSTCFLAGS:=$(filter-out -O2,$(HOSTCFLAGS))

else

obj-y += origen.o

endif

改为：

obj-y += fs4412.o

* 1. 修改include/configs/fs4412.h

将：

#ifndef \_\_CONFIG\_ORIGEN\_H

#define \_\_CONFIG\_ORIGEN\_H

改为：

#ifndef \_\_CONFIG\_FS4412\_H

#define \_\_CONFIG\_FS4412\_H

将：

#define CONFIG\_EXYNOS4210 1

#define CONFIG\_ORIGEN 1

改为：

#define CONFIG\_EXYNOS4412 1

#define CONFIG\_FS4412 1

删除：

/\* MMC SPL \*/

#define COPY\_BL2\_FNPTR\_ADDR 0x02020030

#define CONFIG\_SPL\_TEXT\_BASE 0x02021410

#define CONFIG\_EXTRA\_ENV\_SETTINGS \

"loadaddr=0x40007000\0" \

"rdaddr=0x48000000\0" \

"kerneladdr=0x40007000\0" \

"ramdiskaddr=0x48000000\0" \

"console=ttySAC2,115200n8\0" \

"mmcdev=0\0" \

"bootenv=uEnv.txt\0" \

"loadbootenv=load mmc ${mmcdev} ${loadaddr} ${bootenv}\0" \

"importbootenv=echo Importing environment from mmc ...; " \

"env import -t $loadaddr $filesize\0" \

"loadbootscript=load mmc ${mmcdev} ${loadaddr} boot.scr\0" \

"bootscript=echo Running bootscript from mmc${mmcdev} ...; " \

"source ${loadaddr}\0"

#define CONFIG\_BOOTCOMMAND \

"if mmc rescan; then " \

"echo SD/MMC found on device ${mmcdev};" \

"if run loadbootenv; then " \

"echo Loaded environment from ${bootenv};" \

"run importbootenv;" \

"fi;" \

"if test -n $uenvcmd; then " \

"echo Running uenvcmd ...;" \

"run uenvcmd;" \

"fi;" \

"if run loadbootscript; then " \

"run bootscript; " \

"fi; " \

"fi;" \

"load mmc ${mmcdev} ${loadaddr} uImage; bootm ${loadaddr} "

将：

#define CONFIG\_IDENT\_STRING " for ORIGEN"

改为：

#define CONFIG\_IDENT\_STRING " for FS4412"

删除：

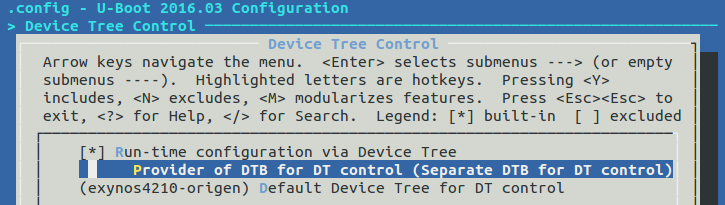
#define CONFIG\_SPL\_LDSCRIPT "board/samsung/common/exynos-uboot-spl.lds"

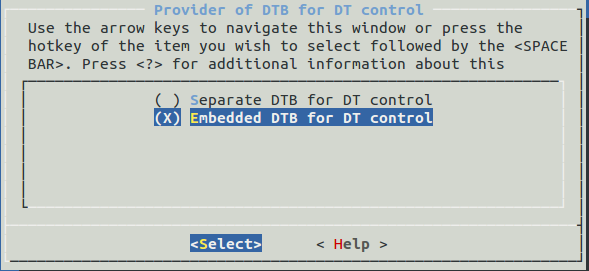
#define CONFIG\_SPL\_MAX\_FOOTPRINT (14 \* 1024)

1. 配置U-Boot。

$ make fs4412\_defconfig

$ make menuconfig





1. 编译U-Boot。

$ make CROSS\_COMPILE=arm-none-linux-gnueabi-

1. 测试U-Boot。

将生成的u-boot.bin文件拷贝到TFTP服务器指定的下载目录。

$ sudo cp u-boot.bin /var/lib/tftpboot/

开发板启动后，在U-Boot的命令行中执行以下的命令来测试新编译的U-Boot。

FS4412 # tftpboot 43E00000 u-boot.bin

FS4412 # go 43E00000

运行的效果如下：

FS4412 # tftpboot 43E00000 u-boot.bin

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: e2:33:0d:71:3c:f7

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'u-boot.bin'.

Load address: 0x43e00000

Loading: ########################

610.4 KiB/s

done

Bytes transferred = 338396 (529dc hex)

FS4412 # go 43E00000

## Starting application at 0x43E00000 ...

U-Boot 2016.03 (Apr 30 2016 - 08:10:56 +0800) for FS4412

CPU: Exynos4412 @ 1 GHz

Model: FS4412 based on Exynos4412

Board: FS4412 based on Exynos4412

DRAM: 1 GiB

WARNING: Caches not enabled

MMC: DWMMC56: Can't get bus-width

exynos\_dwmci\_process\_node: failed to decode dev 0

SAMSUNG SDHCI: 0

\*\*\* Warning - read failed, using default environment

FS4412 #

# 实验十一 U-Boot网卡驱动移植

**【实验目的】**

熟悉U-Boot网卡驱动移植过程。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改include/configs/fs4412.h

删除：

#undef CONFIG\_CMD\_PING

在文件的最后的#endif前添加：

#ifdef CONFIG\_CMD\_NET

#define CONFIG\_NET\_MULTI

#define CONFIG\_DRIVER\_DM9000 1

#define CONFIG\_DM9000\_BASE 0x05000000

#define DM9000\_IO CONFIG\_DM9000\_BASE

#define DM9000\_DATA (CONFIG\_DM9000\_BASE + 4)

#define CONFIG\_DM9000\_USE\_16BIT

#define CONFIG\_DM9000\_NO\_SROM 1

#define CONFIG\_IPADDR 192.168.10.200

#define CONFIG\_SERVERIP 192.168.10.100

#define CONFIG\_GATEWAYIP 192.168.10.1

#define CONFIG\_NETMASK 255.255.255.0

#endif

#endif /\* \_\_CONFIG\_H \*/

1. 修改board/samsung/common/board.c

在：

s5p\_config\_sromc(config.bank, smc\_bw\_conf, smc\_bc\_conf);

return smc911x\_initialize(0, base\_addr);

#endif

后添加：

#ifdef CONFIG\_DRIVER\_DM9000

return dm9000\_initialize(bis);

#endifs’s’

1. 配置U-Boot。

Command line interface --->

Network commands --->

[\*] bootp, tftpboot

-\*- Networking support --->

[\*] Random ethaddr if unset

1. 编译U-Boot。

$ make CROSS\_COMPILE=arm-none-linux-gnueabi-

1. 测试U-Boot。

将生成的u-boot.bin文件拷贝到TFTP服务器指定的下载目录。

$ sudo cp u-boot.bin /var/lib/tftpboot/

开发板启动后，在U-Boot的命令行中执行以下的命令来测试新编译的U-Boot。

FS4412 # tftpboot 43E00000 u-boot.bin

FS4412 # go 43E00000

FS4412 # ping 192.168.10.100

FS4412 # tftpboot 43E00000 u-boot.bin

U-Boot 2016.03 (Apr 30 2016 - 09:02:23 +0800) for FS4412

CPU: Exynos4412 @ 1 GHz

Model: FS4412 based on Exynos4412

Board: FS4412 based on Exynos4412

DRAM: 1 GiB

WARNING: Caches not enabled

MMC: DWMMC56: Can't get bus-width

exynos\_dwmci\_process\_node: failed to decode dev 0

SAMSUNG SDHCI: 0

mmc fail to send stop cmd

\*\*\* Warning - read failed, using default environment

Net: dm9000

Warning: dm9000 (eth0) using random MAC address - d2:ef:59:ca:cb:25

FS4412 #

FS4412 # ping 192.168.10.100

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: d2:ef:59:ca:cb:25

operating at 100M full duplex mode

Using dm9000 device

host 192.168.10.100 is alive

FS4412 # tftpboot 43E00000 u-boot.bin

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: d2:ef:59:ca:cb:25

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'u-boot.bin'.

Load address: 0x43e00000

Loading: #########################

634.8 KiB/s

done

Bytes transferred = 365820 (594fc hex)

# 实验十二 U-Boot SD/eMMC驱动移植

**【实验目的】**

熟悉U-Boot网卡驱动移植过程。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改arch/arm/dts/exynos4412-fs4412.dts

将：

aliases {

serial0 = "/serial@13800000";

console = "/serial@13820000";

mmc2 = "sdhci@12530000";

};

改为：

aliases {

console = "/serial@13820000";

mmc2 = "sdhci@12530000";

mmc4 = "dwmmc@12550000";

};

serial@13820000 {

status = "okay";

};

将：

sdhci@12530000 {

samsung,bus-width = <4>;

samsung,timing = <1 2 3>;

cd-gpios = <&gpk2 2 0>;

};

改为：

sdhci@12530000 {

samsung,bus-width = <4>;

samsung,timing = <0 0 0>;

cd-gpios = <&gpx0 7 0>;

pwr-gpios = <&gpk0 2 0>;

};

添加：

dwmmc@12550000 {

samsung,bus-width = <8>;

samsung,timing = <2 1 0>;

samsung,removable = <0>;

fifoth\_val = <0x203f0040>;

bus\_hz = <400000000>;

div = <0x3>;

index = <4>;

};

1. 修改include/configs/fs4412.h

将：

#define RESERVE\_BLOCK\_SIZE (512)

#define BL1\_SIZE (16 << 10) /\*16 K reserved for BL1\*/

#define CONFIG\_ENV\_OFFSET (RESERVE\_BLOCK\_SIZE + BL1\_SIZE)

改为：

#define BL1\_SIZE (8 << 10) /\* 8 K reserved for BL1 \*/

#define BL2\_SIZE (16 << 10) /\* 16 K reserved for BL2 \*/

#define U\_BOOT\_SIZE (472 << 10) /\* 472 K reserved for BL2 \*/

#define CONFIG\_ENV\_OFFSET (BL1\_SIZE + BL2\_SIZE + U\_BOOT\_SIZE) /\* BL1 + BL2 + U-BOOT + ENV = 512K \*/

删除：

/\* U-Boot copy size from boot Media to DRAM.\*/

#define COPY\_BL2\_SIZE 0x80000

#define BL2\_START\_OFFSET ((CONFIG\_ENV\_OFFSET + CONFIG\_ENV\_SIZE)/512)

#define BL2\_SIZE\_BLOC\_COUNT (COPY\_BL2\_SIZE/512)

在删除的位置添加：

#define CONFIG\_SUPPORT\_EMMC\_BOOT

1. 编译U-Boot。

$ make CROSS\_COMPILE=arm-none-linux-gnueabi-

1. 测试U-Boot。

将生成的u-boot.bin文件拷贝到TFTP服务器指定的下载目录。

$ sudo cp u-boot.bin /var/lib/tftpboot/

开发板启动后，在U-Boot的命令行中执行以下的命令来测试新编译的U-Boot。

FS4412 # tftpboot 43E00000 u-boot.bin

FS4412 # go 43E00000

FS4412 # mmc list

mmc的测试可以参考实验七的命令使用。

1. 测试通过后，使用下面的命令将U-Boot的当前配置文件保存为默认的配置文件，方便之后可以使用该默认配置。

$ cp .config configs/fs4412\_defconfig

$ make clean

$ make distclean

# 实验十三 sdtool工具的使用

**【实验目的】**

熟悉sdtool工具的使用方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. U-Boot移植完成后，需要将镜像同bl1、bl2合并，需要使用sdtool工具。这是一个脚本，用法如下：
   1. 进入到sdtool目录
   2. $ ./sdtool.sh clean # 清除所有目标文件
   3. $ ./sdtool.sh update # 重新生成目标文件
   4. $ ./sdtool.sh mkuboot u-boot.bin # 将bl1、bl2、u-boot.bin合并，生成U-Boot的烧写镜像u-boot-fs4412.bin
   5. $ sudo ./sdtool.sh fuse /dev/sdb u-boot-fs4412.bin # 将镜像文件烧写到/dev/sdb设备上，一般是SD卡

# 实验十四 Linux内核的配置和编译

**【实验目的】**

掌握Linux内核的配置和编译的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 将“系统移植\3-案例源码\第三天\[linux-3.14.25.tar.xz](https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.14.25.tar.xz)”拷贝到Ubuntu主机。
2. 使用下面的命令对源码进行解压。

$ tar -xvf linux-3.14.25.tar.xz

1. 进入到解压的目录，清除配置。

$ cd linux-3.14.25/

$ make ARCH=arm mrproper

或

$ make ARCH=arm distclean

1. 使用下面的命令可以对内核进行exynos的默认配置。

make ARCH=arm exynos\_defconfig

也可以直接拷贝默认的配置文件，但在编译前必须要进行一次make menuconfig。

$ cp arch/arm/configs/exynos\_defconfig .config

1. 使用下面的命令，对内核进行配置。

$ make ARCH=arm menuconfig

指定交叉编译工具链的前缀

General setup --->

(arm-none-linux-gnueabi-) Cross-compiler tool prefix

1. 编译内核的uImage镜像。

$ make ARCH=arm uImage

或

$ make ARCH=arm uImage -j2

或

$ make ARCH=arm uImage -j2 > /dev/null

其中j后面的数字通常是CPU个数的两倍。

如果不想每次都在命令行中输入ARCH=arm，则可以通过修改顶层的Makefile文件来实现。

将：

ARCH ?= $(SUBARCH)

改为：

ARCH ?= arm

1. 生成设备树文件。

$ cp arch/arm/boot/dts/exynos4412-origen.dts arch/arm/boot/dts/exynos4412-fs4412.dts

$ vim arch/arm/boot/dts/Makefile

将：

dtb-$(CONFIG\_ARCH\_EXYNOS) += exynos4210-origen.dtb \

exynos4210-smdkv310.dtb \

……

exynos4412-origen.dtb \

……

exynos5440-ssdk5440.dtb

添加一行exynos4412-fs4412.dtb的内容，变为如下：

dtb-$(CONFIG\_ARCH\_EXYNOS) += exynos4210-origen.dtb \

exynos4210-smdkv310.dtb \

……

exynos4412-origen.dtb \

exynos4412-fs4412.dtb \

……

exynos5440-ssdk5440.dtb

最后使用如下命令编译设备树文件。

$ make ARCH=arm dtbs

1. 拷贝内核和设备树文件到TFTP服务器指定的目录。

$ sudo cp arch/arm/boot/uImage /var/lib/tftpboot/

$ sudo cp arch/arm/boot/dts/exynos4412-fs4412.dtb /var/lib/tftpboot/

1. 按照实验八的方法测试内核，发现内核并不能正常启动。输出的信息如下。

FS4412 # boot

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 72:7e:05:6d:5a:58

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'uImage'.

Load address: 0x41000000

Loading: #################################################################

#################################################################

###########################################################

613.3 KiB/s

done

Bytes transferred = 2767248 (2a3990 hex)

dm9000 i/o: 0x5000000, id: 0x90000a46

DM9000: running in 16 bit mode

MAC: 72:7e:05:6d:5a:58

operating at 100M full duplex mode

Using dm9000 device

TFTP from server 192.168.10.100; our IP address is 192.168.10.200

Filename 'exynos4412-fs4412.dtb'.

Load address: 0x42000000

Loading: ###

545.9 KiB/s

done

Bytes transferred = 33546 (830a hex)

## Booting kernel from Legacy Image at 41000000 ...

Image Name: Linux-3.14.25

Image Type: ARM Linux Kernel Image (uncompressed)

Data Size: 2767184 Bytes = 2.6 MiB

Load Address: 40008000

Entry Point: 40008000

Verifying Checksum ... OK

## Flattened Device Tree blob at 42000000

Booting using the fdt blob at 0x42000000

Loading Kernel Image ... OK

Loading Device Tree to 4fff4000, end 4ffff309 ... OK

Starting kernel ...

# 实验十五 Linux内核网卡驱动移植

**【实验目的】**

熟悉Linux内核网卡驱动移植的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改arch/arm/boot/dts/exynos4412-fs4412.dts

删除设备节点：

firmware@0203F000 {

compatible = "samsung,secure-firmware";

reg = <0x0203F000 0x1000>;

};

添加设备节点：

srom-cs1@5000000 {

compatible = "simple-bus";

#address-cells = <1>;

#size-cells = <1>;

reg = <0x5000000 0x1000000>;

ranges;

ethernet@5000000 {

compatible = "davicom,dm9000";

reg = <0x5000000 0x2 0x5000004 0x2>;

interrupt-parent = <&gpx0>;

interrupts = <6 4>;

davicom,no-eeprom;

mac-address = [00 0a 2d a6 55 a2];

};

};

1. 修改drivers/clk/clk.c

将529行的：

static bool clk\_ignore\_unused;

改为：

static bool clk\_ignore\_unused = true;

否则要在bootargs中指定。

1. 配置内核。

添加驱动支持。

Device Drivers --->

[\*] Network device support --->

[\*] Ethernet driver support --->

<\*> DM9000 support

[ ] Force simple NSR based PHY polling (NEW)

添加网络支持。

[\*] Networking support --->

Networking options --->

<\*> Packet socket

<\*> Unix domain sockets

<\*> PF\_KEY sockets

[\*] TCP/IP networking

[\*] IP: kernel level autoconfiguration

添加文件系统支持。

File systems --->

[\*] Network File Systems --->

<\*> NFS client support

<\*> NFS client support for NFS version 2 (NEW)

<\*> NFS client support for NFS version 3 (NEW)

[\*] Root file system on NFS

1. 重新编译、拷贝镜像和测试内核，输出信息如下。系统正常启动。

Starting kernel ...

[ 0.000000] Booting Linux on physical CPU 0xa00

[ 0.000000] Linux version 3.14.25 (farsight@ubuntu) (gcc version 4.6.4 (crosstool-NG hg+default-2685dfa9de14 - tc0002) ) #2 SMP PREEMPT Sat Apr 30 11:49:13 CST 2016

[ 0.000000] CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=10c5387d

[ 0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache

……

[ 2.085000] devtmpfs: mounted

[ 2.090000] Freeing unused kernel memory: 228K (c0511000 - c054a000)

# 实验十六 Linux内核SD/eMMC驱动移植

**【实验目的】**

熟悉Linux内核SD/eMMC驱动移植的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改arch/arm/boot/dts/exynos4412-fs4412.dts

将：

sdhci@12530000 {

bus-width = <4>;

pinctrl-0 = <&sd2\_clk &sd2\_cmd &sd2\_bus4 &sd2\_cd>;

pinctrl-names = "default";

vmmc-supply = <&mmc\_reg>;

status = "okay";

};

改为：

sdhci@12530000 {

bus-width = <4>;

pinctrl-0 = <&sd2\_clk &sd2\_cmd &sd2\_bus4>;

pinctrl-names = "default";

cd-gpios = <&gpx0 7 0>;

cd-inverted = <0>;

status = "okay";

};

1. 配置内核。

添加驱动支持。

Device Drivers --->

<\*> MMC/SD/SDIO card support --->

<\*> Secure Digital Host Controller Interface support

<\*> SDHCI support on Samsung S3C SoC

添加文件系统及语言支持。

File systems --->

DOS/FAT/NT Filesystems --->

<\*> MSDOS fs support

<\*> VFAT (Windows-95) fs support

(iso8859-1) Default iocharset for FAT

-\*- Native language support --->

<\*> Codepage 437 (United States, Canada)

<\*> Simplified Chinese charset (CP936, GB2312)

<\*> ASCII (United States)

<\*> NLS ISO 8859-1 (Latin 1; Western European Languages)

<\*> NLS UTF-8

1. 测试内核。按照上一实验的方法测试内核，SD卡和eMMC设备被正确探测。

[ 1.730000] mmc1: new high speed DDR MMC card at address 0001

[ 1.735000] mmcblk0: mmc1:0001 4YMD3R 3.64 GiB

[ 1.740000] mmcblk0boot0: mmc1:0001 4YMD3R partition 1 4.00 MiB

[ 1.745000] mmcblk0boot1: mmc1:0001 4YMD3R partition 2 4.00 MiB

[ 1.750000] mmcblk0rpmb: mmc1:0001 4YMD3R partition 3 512 KiB

[ 1.755000] mmcblk0: p1

[ 1.760000] mmcblk0boot1: unknown partition table

[ 1.765000] mmcblk0boot0: unknown partition table

[ 1.850000] mmc0: new high speed SDHC card at address 0001

[ 1.855000] mmcblk1: mmc0:0001 00000 7.44 GiB

[ 1.860000] mmcblk1: p1

# 实验十七 Linux内核USB主机控制器驱动移植

**【实验目的】**

熟悉Linux内核USB主机控制器驱动移植的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改arch/arm/boot/dts/exynos4412-fs4412.dts

添加：

usbphy: usbphy@125B0000 {

#address-cells = <1>;

#size-cells = <1>;

compatible = "samsung,exynos4x12-usb2phy";

reg = <0x125B0000 0x100>;

ranges;

clocks = <&clock 2>, <&clock 305>;

clock-names = "xusbxti", "otg";

status = "okay";

usbphy-sys {

reg = <0x10020704 0x8>;

};

};

ehci@12580000 {

usbphy = <&usbphy>;

status = "okay";

};

usb3503@08 {

compatible = "smsc,usb3503";

reg = <0x08 0x4>;

connect-gpios = <&gpm3 3 1>;

intn-gpios = <&gpx2 3 1>;

reset-gpios = <&gpm2 4 1>;

initial-mode = <1>;

};

1. 配置内核。

添加驱动支持。

Device Drivers --->

[\*] USB support --->

<\*> EHCI HCD (USB 2.0) support

<\*> EHCI support for Samsung S5P/EXYNOS SoC Series

<\*> USB Mass Storage support

<\*> USB3503 HSIC to USB20 Driver

USB Physical Layer drivers --->

<\*> Samsung USB 2.0 PHY controller Driver

SCSI device support --->

<\*> SCSI device support

<\*> SCSI disk support

1. 重新编译设备树和内核，在开发板上接上U盘，启动内核会发现U盘被正确识别。

[ 1.975000] usb 1-3: new high-speed USB device number 2 using exynos-ehci

[ 2.115000] hub 1-3:1.0: USB hub found

[ 2.115000] hub 1-3:1.0: 3 ports detected

[ 2.415000] usb 1-3.1: new high-speed USB device number 3 using exynos-ehci

[ 2.525000] usb-storage 1-3.1:1.0: USB Mass Storage device detected

[ 2.530000] scsi0 : usb-storage 1-3.1:1.0

[ 3.535000] scsi 0:0:0:0: Direct-Access Generic USB SD Reader 1.00 PQ: 0 ANSI: 0 CCS

[ 3.555000] sd 0:0:0:0: [sda] 15771648 512-byte logical blocks: (8.07 GB/7.52 GiB)

[ 3.560000] sd 0:0:0:0: [sda] Write Protect is off

[ 3.565000] sd 0:0:0:0: [sda] No Caching mode page found

[ 3.570000] sd 0:0:0:0: [sda] Assuming drive cache: write through

[ 3.580000] sd 0:0:0:0: Attached scsi generic sg0 type 0

[ 3.590000] sd 0:0:0:0: [sda] No Caching mode page found

[ 3.595000] sd 0:0:0:0: [sda] Assuming drive cache: write through

[ 3.600000] sda: sda1

[ 3.620000] sd 0:0:0:0: [sda] No Caching mode page found

[ 3.625000] sd 0:0:0:0: [sda] Assuming drive cache: write through

[ 3.630000] sd 0:0:0:0: [sda] Attached SCSI removable disk

在根文件系统挂载成功后，执行下面的命令可以把U盘的第一个分区挂载到/mnt目录下。

# mount -t vfat /dev/sda1 /mnt

# echo "FS4412" > /mnt/test.txt

# cat /mnt/test.txt

FS4412

# umount /mnt

# 实验十八 Linux内核LCD驱动移植

**【实验目的】**

熟悉Linux内核USB主机控制器驱动移植的方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，修改arch/arm/boot/dts/exynos4412-fs4412.dts

将：

fimd@11c00000 {

pinctrl-0 = <&lcd\_clk &lcd\_data24 &pwm1\_out>;

pinctrl-names = "default";

status = "okay";

};

display-timings {

native-mode = <&timing0>;

timing0: timing {

clock-frequency = <47500000>;

hactive = <1024>;

vactive = <600>;

hfront-porch = <64>;

hback-porch = <16>;

hsync-len = <48>;

vback-porch = <64>;

vfront-porch = <16>;

vsync-len = <3>;

};

};

改为：

fimd: fimd@11c00000 {

pinctrl-0 = <&lcd\_clk &lcd\_data24 &pwm1\_out>;

pinctrl-names = "default";

status = "okay";

display-timings {

native-mode = <&timing0>;

timing0: timing {

hsync-active = <0>;

vsync-active = <0>;

de-active = <0>;

pixelclk-active = <1>;

clock-frequency = <51206400>;

hactive = <1024>;

vactive = <600>;

hfront-porch = <150>;

hback-porch = <160>;

hsync-len = <10>;

vback-porch = <22>;

vfront-porch = <12>;

vsync-len = <1>;

};

};

};

1. 编写drivers/gpu/drm/exynos/exynos\_drm\_fbclk.h和drivers/gpu/drm/exynos/exynos\_drm\_fbclk.c（源码位于“案例源码\第三天”，可直接拷贝），内容如下：

/\* drivers/gpu/drm/exynos/exynos\_drm\_fbclk.h \*/

#ifndef \_EXYNOS\_DRM\_FBCLK\_H\_

#define \_EXYNOS\_DRM\_FBCLK\_H\_

void exynos\_drm\_fbclk\_preinit(void);

#endif

/\* drivers/gpu/drm/exynos/exynos\_drm\_fbclk.c \*/

#include <linux/io.h>

#include <linux/ioport.h>

#include <mach/map.h>

#define CLK\_SRC\_LCD0 (S5P\_VA\_CMU + 0xC234)

#define CLK\_SRC\_MASK\_LCD (S5P\_VA\_CMU + 0xC334)

#define CLK\_DIV\_LCD (S5P\_VA\_CMU + 0xC534)

#define CLK\_DIV\_STAT\_LCD (S5P\_VA\_CMU + 0xC634)

#define CLK\_GATE\_IP\_LCD (S5P\_VA\_CMU + 0xC934)

#define CLK\_GATE\_BLOCK (S5P\_VA\_CMU + 0xC970)

#define LCDBLK\_CFG (S3C\_VA\_SYS + 0x0210)

void exynos\_drm\_fbclk\_preinit(void)

{

/\* FIMD0\_SEL = SCLKVPLL, 350000000 \*/

\_\_raw\_writel((\_\_raw\_readl(CLK\_SRC\_LCD0) & ~(0x0F)) | (0x08), CLK\_SRC\_LCD0);

/\* FIMD0\_RATIO = 3, SCLK\_FIMD0 = MOUTFIMD0/(FIMD0\_RATIO + 1) \*/

\_\_raw\_writel((\_\_raw\_readl(CLK\_DIV\_LCD) & ~(0x0F)) | (0x04), CLK\_DIV\_LCD);

/\* unmask output clock of MUXFIMD0 \*/

\_\_raw\_writel(\_\_raw\_readl(CLK\_SRC\_MASK\_LCD) | 0x1, CLK\_SRC\_MASK\_LCD);

\_\_raw\_writel(\_\_raw\_readl(CLK\_GATE\_IP\_LCD) | 0x1, CLK\_GATE\_IP\_LCD);

\_\_raw\_writel(\_\_raw\_readl(CLK\_GATE\_BLOCK) | (0x1 << 4), CLK\_GATE\_BLOCK);

/\* select fimd \*/

\_\_raw\_writel(\_\_raw\_readl(LCDBLK\_CFG) | (0x1 << 1), LCDBLK\_CFG);

}

1. 修改drivers/gpu/drm/exynos/Makefile

将：

exynosdrm-$(CONFIG\_DRM\_EXYNOS\_FIMD)+= exynos\_drm\_fimd.o

改为：

exynosdrm-$(CONFIG\_DRM\_EXYNOS\_FIMD)+= exynos\_drm\_fimd.o exynos\_drm\_fbclk.o

1. 修改drivers/gpu/drm/exynos/exynos\_drm\_drv.c

添加头文件：

#include "exynos\_drm\_iommu.h"

#include "exynos\_drm\_fbclk.h"

将：

/\*

\* create and configure fb helper and also exynos specific

\* fbdev object.

\*/

ret = exynos\_drm\_fbdev\_init(dev);

if (ret) {

DRM\_ERROR("failed to initialize drm fbdev\n");

goto err\_drm\_device;

}

改为：

/\*

\* create and configure fb helper and also exynos specific

\* fbdev object.

\*/

exynos\_drm\_fbclk\_preinit();

ret = exynos\_drm\_fbdev\_init(dev);

if (ret) {

DRM\_ERROR("failed to initialize drm fbdev\n");

goto err\_drm\_device;

}

即添加exynos\_drm\_fbclk\_preinit函数调用。

1. 添加驱动支持。

Device Drivers --->

Graphics support --->

<\*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) ----

<\*> DRM Support for Samsung SoC EXYNOS Series

[\*] Exynos DRM FIMD

1. 重新编译设备树和内核，并测试内核，发现LCD屏上出现了4个小企鹅。

# 实验十九 LED驱动移植

**【实验目的】**

驱动程序的编译。

应用程序如何打开/操作设备

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 添加驱动文件

将实验代码Led\_test/fs4412\_led\_drv.c拷贝到drivers/char下。

1. 修改drivers/char/Kconfig，在menu "Character devices"下面添加如下内容：

config FS4412\_LED

tristate "FS4412LED Device Support"

depends onARCH\_EXYNOS4

help

support leddevice on FS4412develop board

1. 修改drivers/char/Makefile，在文件最后添加如下代码

obj-$(CONFIG\_FS4412\_LED) += fs4412\_led\_drv.o

1. 将s5pv210\_led\_app.c拷贝到Linux任意目录下并交叉编译测试程序

$ arm-none-linux-gnueabi-gccfs4412\_led\_app.c –o fs4412\_led\_app

sudo cpfs4412\_led\_app /source/rootfs

1. 编译LED驱动到内核中

* 配置内核时按“空格”选择，配置完成后保存退出

$ make menuconfig

Device Drivers --->

Character devices --->

<\*>FS4412 LED Device Support

* 保存退出，重新编译后把uImage拷贝到tftpboot下

$ make uImage

$ cp arch/arm/boot/uImage/tftpboot

重新启动开发板，加载内核并运行。在终端下执行下面操作

* 创建设备节点

# mknod /dev/ledc5000

* 运行测试程序并观察现象

# ./fs4412\_led\_app

1. 编译LED驱动为模块

* 配置内核时按“空格”选择，配置完成后保存退出

$ make menuconfig

Device Drivers --->

Character devices --->

<M>FS4412 LED Device Support

* 保存退出，重新编译后把zImage拷贝到tftpboot下，把驱动模块拷贝到/source/rootfs下

$ make uImage modules

$ cp arch/arm/boot/uImage /tftpboot/

$ cp drivers/char/fs4412\_led\_drv.ko /source/rootfs/

重新启动开发板，linux运行起来后在终端下操作

* 创建设备节点

# mknod /dev/ledc5000

* 加载LED驱动模块

#insmod fs4412\_led\_drv.ko

* 运行测试程序并观察现象

# ./fs4412\_led\_app

# 实验二十内存调试

**【实验目的】**

段错误和内存错误是C语言编程经常遇到的问题，使用memwatch是由johan lindh编写，是一个开发源代码C语言内存错误检测工具。能检测双重释放，错误释放，没有释放内存，溢出等等情况。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 解压memwatch.-2.7.1.tar.gz，在解开的目录下添加代码memtest.c

#include <stdlib.h>

#include <stdio.h>

#include "memwatch.h"

int main(int argc,char \*\*argv)

{

char \*ptr1;

char \*ptr2;

ptr1 = malloc(512);

ptr2 = malloc(512);

ptr1[512]= 'A';

ptr2 = ptr1;

free(ptr2);

free(ptr1);

return 0;

}

1. 修改Makefile

Makefile文件为：

memtest:

$(CC) -DMEMWATCH -DMW\_STDIO memtest.cmemwatch.c

1. 运行make并在主机上执行a.out，执行后会生成一个记录文件memwatch.log，内容如下：

============= MEMWATCH 2.71 Copyright (C) 1992-1999 Johan Lindh =========

Started at Thu Jan 1 00:08:33 1970

Modes: \_\_STDC\_\_ 32-bit mwDWORD==(unsigned long)

mwROUNDALLOC==4 sizeof(mwData)==32 mwDataSize==32

overflow: <3> memtest.c(12), 512 bytes alloc'd at <1> memtest.c(8)

double-free: <4> memtest.c(13), 0x1a1b4 was freed from memtest.c(12)

Stopped at Thu Jan 1 00:08:33 1970

unfreed: <2> test.c(9), 512 bytes at 0x1a3e4 {FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE ................}

Memory usage statistics (global):

N)umber of allocations made: 2

L)argest memory usage : 1024

T)otal of all alloc() calls: 1024

U)nfreed bytes totals : 512

//overflow: <3> memtest.c(12)缓冲区溢出，当程序执行到第12行free(ptr2)才检测到的；

512 bytes alloc'd at <1>memtest.c(8)

//表示出错缓冲区的大小为512字节，是在memtest.c的第8行分配的。很容易发现代码的ptr1[512]='A'出现错误。

double-free: <4> memtest.c(13), 0x1a7f4 was freed from memtest.c(12)

//double-free: <4> memtest.c(13)是一个双重释放的错误，表示程序执行到13行的时候才检测到。

0x1a7f4 was freed from memtest.c(12)

//表示首地址为 0x1a7f4的内存在12行已经被释放。

Stopped at Wed Dec 31 19:00:38 1969

unfreed: <2> test.c(9), 512 bytes at 0x1a3e4 {FE FE FE FE FE FE FE FE

//表示一块内存没有释放，表示这块内存是在memtest.c的第9行分配，大小为512字节，首地址为0x1a3e4。

Memory usage statistics (global):

N)umber of allocations made: 2

L)argest memory usage : 1024

//程序结束时能够是使用的最大动态内存

T)otal of all alloc() calls: 1024 //总共分配的动态内存

U)nfreed bytes totals : 512

//表示未释放的内存

# 实验二十一内核调试

**【实验目的】**

由于在驱动开发中经常看到内核崩溃的问题，最常见的就是OOPS错误，本实验要求学员掌握这种调试方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 通过OOPS信息中PC寄存器的值可以知道出错指令的地址，通过栈回朔信息可以知道出错时的函数调用的关系，根据这两点可以很快定位错误。
2. 修改drivers/net/ethernet/davicom/dm9000.c，在dm9000\_probe函数中u32 id\_val;下增加下面语句：

int \*ptr =NULL;

\*ptr=0xff;

1. 编译内核下载到开发板上，内核启动会出现如类似下信息：

Unable to handle kernel NULL pointer dereference at virtual address 00000000

pgd = c0004000

[00000000] \*pgd=00000000

Internal error: Oops: 805 [#1] PREEMPT SMP ARM

Modules linked in:

CPU: 1 PID: 1 Comm: swapper/0 Not tainted 3.14.0 #11

task: ee8a0000 ti: ee8a4000 task.ti: ee8a4000

PC is at dm9000\_probe+0x1c/0x8f0

LR is at platform\_drv\_probe+0x18/0x48

pc : [<c0277cc8>] lr : [<c0247f7c>] psr: 60000153

sp : ee8a5e48 ip : 00000000 fp : 00000000

r10: c052a4fc r9 : 00000000 r8 : c0591e98

r7 : 00000000 r6 : ee97c810 r5 : ee97c800 r4 : 00000000

r3 : 000000ff r2 : 00000000 r1 : ee8a5de8 r0 : ee97c800

Flags: nZCv IRQs on FIQs off Mode SVC\_32 ISA ARM Segment kernel

Control: 10c5387d Table: 4000404a DAC: 00000015

Process swapper/0 (pid: 1, stack limit = 0xee8a4240)

Stack: (0xee8a5e48 to 0xee8a6000)

5e40: ee975cf0 00000000 ee1503a8 00000001 c0561afc ee150438

5e60: 00000000 ee97c810 c0591e98 ee97c810 00000000 c0591e98 c0561afc c052a4fc

5e80: 00000000 c0247f7c c0247f64 c05d931c c0591e98 c0246668 ee97c810 c0591e98

5ea0: ee97c844 00000000 c054332c c0246804 c0591e98 c0246778 00000000 c0244fbc

5ec0: ee805478 ee9771c0 c0591e98 eeb73a00 c0590028 c0245e28 c04c3128 c0591e98

5ee0: 00000000 c0591e98 00000000 c054e2ac c059f280 c0246e1c 00000000 ee8a4000

5f00: 00000000 c00087b4 ee903b00 c05c3d50 60000153 c0571c00 60000100 c0571c00

5f20: 00000000 00000000 c0571bfc 00000000 c0505bc8 ef7fc918 00000089 c0034c6c

5f40: c04ca680 c0505338 00000006 00000006 00000000 c054e2c8 c054e2cc 00000006

5f60: c054e2ac c059f280 00000089 c052a4fc 00000000 c052ac4c 00000006 00000006

5f80: c052a4fc c003e0dc 00000000 c03b46ec 00000000 00000000 00000000 00000000

5fa0: 00000000 c03b46f4 00000000 c000e4b8 00000000 00000000 00000000 00000000

5fc0: 0000000000000000 00000000 00000000 00000000 00000000 00000000 00000000

5fe0: 00000000 00000000 00000000 00000000 00000013 00000000 ffffffff ffffffff

[<c0277cc8>] (dm9000\_probe) from [<c0247f7c>] (platform\_drv\_probe+0x18/0x48)

[<c0247f7c>] (platform\_drv\_probe) from [<c0246668>] (driver\_probe\_device+0x100/0x210)

[<c0246668>] (driver\_probe\_device) from [<c0246804>] (\_\_driver\_attach+0x8c/0x90)

[<c0246804>] (\_\_driver\_attach) from [<c0244fbc>] (bus\_for\_each\_dev+0x58/0x88)

[<c0244fbc>] (bus\_for\_each\_dev) from [<c0245e28>] (bus\_add\_driver+0xd8/0x1cc)

[<c0245e28>] (bus\_add\_driver) from [<c0246e1c>] (driver\_register+0x78/0xf4)

[<c0246e1c>] (driver\_register) from [<c00087b4>] (do\_one\_initcall+0x30/0x144)

[<c00087b4>] (do\_one\_initcall) from [<c052ac4c>] (kernel\_init\_freeable+0xfc/0x1c8)

[<c052ac4c>] (kernel\_init\_freeable) from [<c03b46f4>] (kernel\_init+0x8/0xe4)

1. 错误分析：

1、第一行

Unable to handle kernel NULL pointer dereference at virtual address 00000000

说明是空指针造成的错误

2、寄存器信息主要是PC的值

PC is at dm9000\_probe+0x1c/0x8f0

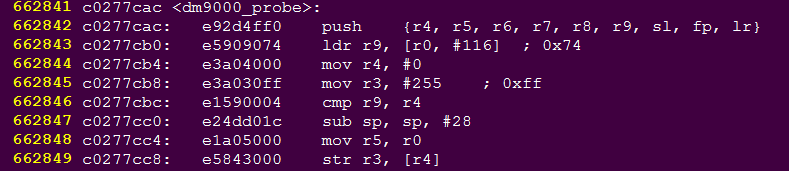
和

pc : [<c0277cc8>]

1. 错误定位：

$ arm-none-linux-gnueabi-objdump -D vmlinux > vmlinux.dis

1. 文件vmlinux.dis非常大打开需要一定时间



1. 对于大多数情况，从反汇编代码定位到C代码并不会如此容易，需要有较强的阅读汇编代码的能力。
2. 另外一种方法是通过addr2line去定位

$ arm-none-linux-gnueabi-addr2line 0xc0277cc8-e vmlinux -f

1. 重要：该实验完成后不要忘记恢复被修改的代码

# 实验二十二制作根文件系统

**【实验目的】**

掌握根文件系统的制作方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 配置并编译Busybox。使用下面的命令。

$wget --no-check-certificate -c http://www.busybox.net/downloads/busybox-1.23.2.tar.bz2 # 源码包在“案例源码\第四天\ busybox-1.23.2.tar.bz2”

$ tar -jxvf busybox-1.23.2.tar.bz2

$ cd busybox-1.23.2/

$ make menuconfig

Busybox Settings --->

Build Options --->

[ ] Build BusyBox as a static binary (no shared libs) (NEW)

(arm-none-linux-gnueabi-) Cross Compiler prefix

Installation Options ("make install" behavior) --->

(/nfs/rootfs) BusyBox installation prefix

Busybox Library Tuning --->

[\*] Username completion

[\*] Fancy shell prompts

1. 编译并安装Busybox。使用下面的命令。

$ make

$ make install # 在install之前，如果/nfs目录下有rootfs目录，先删除。

安装成功后，生成以下目录。

$ ls /nfs/rootfs

bin linuxrc sbin usr

1. 拷贝共享库文件。

$ cd /nfs/rootfs

$ cp -a /home/farsight/gcc-4.6.4/arm-arm1176jzfssf-linux-gnueabi/sysroot/lib/ .

共享库的路径根据交叉编译工具链安装的位置的不同而不同。

$ sudo rm lib/\*.a # 删除静态库文件

$ arm-none-linux-gnueabi-strip lib/\*.so # 裁剪共享库文件，减小体积

1. 安装Linux内核模块，首先进入到移植好的内核源码目录，正确编译完成uImage。然后使用下面的命令编译并安装模块。

$ make ARCH=arm modules

$ sudo make ARCH=arm INSTALL\_MOD\_PATH=/nfs/rootfs modules\_install

1. 添加其他目录。重新进入到rootfs目录，生成相应的目录。

$ cd /nfs/rootfs

$ mkdir dev etc home mnt proc root sys tmp var

$ mkdir etc/init.d

1. 添加/etc/inittab文件，文件内容如下。

::sysinit:/etc/init.d/rcS

::askfirst:-/bin/login

::ctrlaltdel:/sbin/reboot

::shutdown:/sbin/swapoff -a

::shutdown:/bin/umount -a -r

::restart:/sbin/init

1. 添加/etc/ init.d/rcS文件，文件内容如下。

#!/bin/sh

mount -a

mkdir /dev/pts

mount -t devpts devpts /dev/pts

echo /sbin/mdev > /proc/sys/kernel/hotplug

mdev -s

hostname -F /etc/hostname

接下来添加该文件的可执行权限。

$ chmod +x etc/init.d/rcS

1. 添加/etc/fstab文件，文件内容如下。

proc /proc proc defaults 0 0

none /tmp tmpfs defaults 0 0

none /dev tmpfs defaults 0 0

sysfs /sys sysfs defaults 0 0

1. 添加/etc/group文件，文件内容如下。

root:x:0:root

1. 添加/etc/passwd文件，文件内容如下。

root:x:0:0:root:/root:/bin/sh

1. 添加/etc/shadow文件，文件内容如下。

root:BcPgSBqZz80dw:0:0:99999:7:::

1. 添加/etc/hostname文件，文件内容如下。

fs4412

1. 添加/etc/profile文件，文件内容如下。

LD\_LIBRARY\_PATH=/lib:/usr/lib

export LD\_LIBRARY\_PAHT

PATH=/bin:/sbin:/usr/bin:/usr:/sbin

export PATH

export PS1=[\\e[32m[\\u@\\h \\W\\a]\\$\\e[00;37m](file:///\\e[32m[\\u@\\h%20\\W\\a%5d\\$\\e%5b00;37m%20)

1. 按照实验八的方法通过NFS挂载根文件系统。

根文件系统挂载成功后，输出如下信息。

[ 2.175000] hub 1-3:1.0: 3 ports detected

Please press Enter to activate this console.

fs4412 login: root

Password:

login[1214]: root login on 'console'

[root@fs4412 ~]#

按下回车键，然后用户名是root，密码为空，即可登录系统。

# 实验二十三制作ramdisk镜像文件

**【实验目的】**

掌握ramdisk镜像文件的制作方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上，进入到/nfs目录，使用下面的命令制作镜像。

$ cd /nfs/

$ dd if=/dev/zero of=ramdisk bs=1k count=16384

$ mkfs.ext2 -F ramdisk

$ sudo mount -t ext2 ramdisk /mnt

$ sudo cp -a rootfs/\* /mnt/

$ sudo umount /mnt

$ gzip --best -c ramdisk > ramdisk.gz

$ mkimage -n "ramdisk" -A arm -O linux -T ramdisk -C gzip -d ramdisk.gz ramdisk.img

1. 拷贝制作好的镜像文件到TFTP服务器指定的目录。

$ sudo cp ramdisk.img /var/lib/tftpboot/

1. 接下来，重新配置内核，选择ramdisk的支持并指定大小。

File systems --->

<\*> Second extended fs support

Device Drivers

SCSI device support --->

<\*> SCSI disk support

Block devices --->

<\*>RAM block device support

(16)Default number of RAM disks

(16384) Default RAM disk size (kbytes)

General setup --->

[\*] Initial RAM filesystem and RAM disk (initramfs/initrd) support

上面的配置项多数都已配置，主要是修改了ramdisk的大小为16384，这个值必须要和ramdisk镜像文件的大小保持一致。保存配置后，重新编译内核，再将uImage文件拷贝到tftp服务器指定的目录下。

1. 开发板上电，在U-Boot中设置bootcmd环境变量如下。

FS4412 # setenv bootcmd tftp 41000000 uImage\;tftp 42000000 exynos4412-fs4412.dtb\;tftp 43000000 ramdisk.img\;bootm 41000000 43000000 42000000

FS4412 # setenv bootargs

FS4412 # saveenv

最后使用boot命令启动内核并挂载根文件系统，挂载成功后将提示输入用户名和密码（其中用户名为root，密码为空），验证通过后，成功登录系统。输出信息如下：

……

[ 2.250000] VFS: Mounted root (ext2 filesystem) on device 1:0.

Please press Enter to activate this console.

fs4412 login: root

Password:

login[1214]: root login on 'console'

[root@fs4412 ~]#

# 实验二十四烧写内核镜像文件

**【实验目的】**

掌握内核镜像文件的烧写方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 从SD卡启动。
2. 下载uImage到开发板的内存中，并烧写至SD卡。

FS4412 # tftpboot 40000000 uImage

FS4412 # mmc write 40000000 400 3400

1. 下载设备树到开发板的内存中，并烧写至SD卡。

FS4412 # tftpboot 40000000 exynos4412-fs4412.dtb

FS4412 # mmc write 40000000 3800 800

1. 设置环境变量，验证内核镜像烧写成功。

FS4412 # setenv bootcmd mmc read 41000000 400 3400\; mmc read 42000000 3800 800\; bootm 41000000 - 42000000

FS4412 # setenv bootargs noinitrd root=/dev/nfs nfsroot=192.168.10.100:/nfs/rootfs rw console=ttySAC2,115200 init=/linuxrc ip=192.168.10.200

FS4412 # saveenv

FS4412 # boot

1. 重启开发板，从eMMC启动。重复上面的步骤2-步骤4，将内核镜像烧写到eMMC中。

# 实验二十五固化根文件系统到SD/eMMC

**【实验目的】**

掌握内核镜像文件的烧写方法。

说明：在实验中**命令行提示符为“$”表示在主机上运行，“#”表示在目标板上运行。**

**【实验环境】**

1. Ubuntu 12.04发行版
2. FS4412平台

**【实验步骤】**

1. 在上一实验的基础上从SD卡启动，直至系统最终启动成功。
2. 对SD卡进行分区。

[root@fs4412 ~]# fdisk /dev/mmcblk0

注意，如果已有分区，请删除分区。可以用p命令确认。

Command (m for help): p

Disk /dev/mmcblk0: 8010 MB, 8010072064 bytes

4 heads, 16 sectors/track, 244448 cylinders

Units = cylinders of 64 \* 512 = 32768 bytes

Device Boot Start End Blocks Id System

接下来新建一个分区。

Command (m for help): n

Command action

e extended

p primary partition (1-4)

p

Partition number (1-4): 1

First cylinder (1-244448, default 1): 512

Last cylinder or +size or +sizeM or +sizeK (512-244448, default 244448): Using default value 244448

Command (m for help): w

The partition table has been altered.

Calling ioctl() to re-read partition table

[ 34.470000] mmcblk0: p1

上面分区从第512个柱面开始（命令显示每个柱面为32KB），跳过了内核镜像的烧写区域。

1. 格式化SD卡的新建分区。

[root@fs4412 ~]# mkfs.ext2 /dev/mmcblk0p1

1. 拷贝根文件系统到SD卡的新建分区。

[root@fs4412 ~]# mkdir mmc

[root@fs4412 ~]# mount -t ext2 /dev/mmcblk0p1 mmc/

[root@fs4412 ~]# mount -t nfs -o nolock 192.168.10.100:/nfs/rootfs/mnt

[root@fs4412 ~]# cp -a /mnt/\* mmc/

[root@fs4412 ~]# rm -rf mmc/root/mmc/

[root@fs4412 ~]# sync

[root@fs4412 ~]# umount mmc

[root@fs4412 ~]# umount /mnt

[root@fs4412 ~]# rm -rf mmc

1. 重新启动开发板，设置环境变量。

FS4412 # setenv bootargs root=/dev/mmcblk0p1 rw noinitrd console=ttySAC2,115200 init=/linuxrc

FS4412 # saveenv

FS4412 # boot

内核启动后从SD卡挂载根文件系统。

……

[ 1.780000] VFS: Mounted root (ext2 filesystem) on device 179:1.

[ 1.785000] devtmpfs: mounted

[ 1.790000] Freeing unused kernel memory: 232K (c05a7000 - c05e1000)

[ 1.990000] usb 1-3: new high-speed USB device number 2 using exynos-ehci

[ 2.125000] hub 1-3:1.0: USB hub found

[ 2.130000] hub 1-3:1.0: 3 ports detected

Please press Enter to activate this console.

fs4412 login: root

Password:

login[1233]: root login on 'console'

[root@fs4412 ~]#

1. 从eMMC启动。重复步骤2-5，将根文件系统固化在eMMC上。