

专题1-Linux 制作流程

/ 绿色代表引用； 紫色代表重要引用内容； 淡蓝色代表实例代码； 红色代表着重标注，可用于注意事项和重点代码。 */*

一、文件位置

文件名（文件用途）	文件在基础光盘中的路径
uboot1.1.6_FORLINUX_6410.tgz (uboot 源码压缩包)	Linux-3.0.1\uboot_sourcedode\
FORLINUX_linux-3.0.1.tar.gz (Linux-3.0.1 源码压缩包)	Linux-3.0.1\kernel_sourcecode\

二、编译 256M 内存平台 Uboot 方法

进入 uboot1.1.6 目录、配置 config、编译：

#cd uboot1.1.6（进入 uboot 源码的目录）

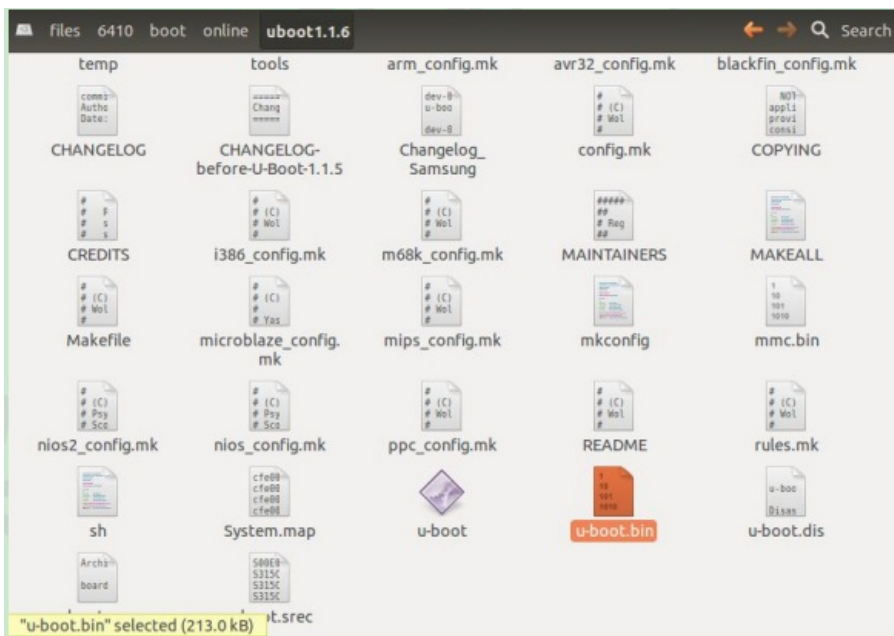
make forlinux_nand_ram256_config（配置适用于 256M 内存平台的 config）

```
root@forlinux: /forlinux/uboot1.16
File Edit View Search Terminal Help
drivers/sk98lin/libsk98lin.a post/libpost.a post/cpu/libcpu.a common/libcommon.a
|sed -n -e 's/.*\(_u boot cmd .*\)/-u\1/p'|sort|uniq`;
cd /forlinux/uboot1.16 && /usr/local/arm/4.3.2/bin/arm-linux-ld -
Bstatic -T /forlinux/uboot1.16/board/samsung/smdk6410/u-boot.lds -Ttext 0xC7E0000
0 $UNDEF_SYM cpu/s3c64xx/start.o \
--start-group lib_generic/libgeneric.a board/samsung/smd
k6410/libsmdk6410.a cpu/s3c64xx/lib3c64xx.a cpu/s3c64xx/s3c6410/lib3c6410.a li
b_arm/libarm.a fs/cramfs/libcramfs.a fs/fat/libfat.a fs/fdos/libfdos.a fs/jffs2/
libjffs2.a fs/reiserfs/libreiserfs.a fs/ext2/libext2fs.a net/libnet.a disk/libdi
sk.a rtc/librtc.a dtb/libdtb.a drivers/libdrivers.a drivers/nand/libnand.a drive
rs/nand_legacy/libnand_legacy.a drivers/onenand/libonenand.a drivers/sk98lin/lib
sk98lin.a post/libpost.a post/cpu/libcpu.a common/libcommon.a --end-group -L /us
r/local/arm/4.3.2/bin/./lib/gcc/arm-none-linux-gnueabi/4.3.2/armv4t -lgcc \
-Map u-boot.map -o u-boot
/usr/local/arm/4.3.2/bin/arm-linux-objcopy --gap-fill=0xff -O srec u-boot u-boot
.srec
/usr/local/arm/4.3.2/bin/arm-linux-objcopy --gap-fill=0xff -O binary u-boot u-bo
ot.bin
/usr/local/arm/4.3.2/bin/arm-linux-objdump -d u-boot > u-boot.dis
root@forlinux:/forlinux/uboot1.16# make forlinux_nand_ram256_config
Configuring for smdk6410 board which boot from NAND ram256...
root@forlinux:/forlinux/uboot1.16#
```

#make clean（删除以前编译的文件）

#make（编译）

如果编译成功，将在 'uboot1.1.6' 目录下产生名为 'u-boot.bin' 的二进制文件。该文件即我们需要烧写到 Nandflash 的 U-boot 映像文件，如下图所示：



三、编译 Linux 系统

命令如下：

```
#cd linux-3.0.1
```

```
#make zImage
```

编译结束后将在内核源码目录的 arch/arm/boot 中得到 Linux 内核映像文件：zImage

注意，假如使用 wifi 功能的话，需要用 make menuconfig 命令对所使用的 wifi 进行下配置再编译内核。配置方法如下：

在命令行输入 make menuconfig，会出现内核配置图形界面，然后依次选择：Device Drivers--->Network device support --->Wireless LAN --->Select rtl wifi，其中 Realtek 8192C USB WiFi 对应使用 rtl8188 芯片的 usb wifi 模块，Realtek 8189E SDIO WiFi 对应飞凌的 8189ES 的 SDIO wifi 模块。选择相应的配置然后点 Select,然后一步步点 “Exit” 退出，最后提示是否保存配置，点 “yes” 配置完成。

四、制作 yaffs2 文件系统映像

文件名（文件用途）	文件在基础光盘中的路径
FileSystem-Yaffs2.tar.gz (文件系统源码压缩包)	Linux-3.0.1\filesystem
mkyaffs2image-nand2g (yaffs2 文件系统映像制作工具，适用于 1G，2G 或者 4G 字节 nandflash)	Linux-3.0.1\filesystem\Yaffs2 文件系统制作工具
mkyaffs2image-nand256m (yaffs2 文件系统映像制作工具，适用于 256m 字节 nandflash)	Linux-3.0.1\filesystem\Yaffs2 文件系统制作工具

4.1、准备好文件系统

这里我们以飞凌提供的文件系统为例，为您演示如何制作 yaffs2 文件系统映像。FileSystem-Yaffs2.tar.gz 是我们提供的文件系统目录，用户可以使用此目录制作 Yaffs2 文件系统，且FileSystem-Yaffs2.tar.gz 目录也用于 NFS 网络根文件系统挂载，NFS 挂载具体参考 4-7-7 节 挂载 NFS网络文件系统。

注意：本次发布的文件系统 FileSystem-Yaffs2.tar.gz 相对于上次文件系统添加了一些有用的qtopia2.2.0 的测试程序，同时也删减了一些 qt4.7.1 的演示程序。将压缩包 ‘FileSystem-Yaffs2.tar.gz’ 及文件系统映像制作工具 mkyaffs2image-nand256m 和mkyaffs2image-nand2g 拷贝到你的工作目录下，解压缩文件系统源码包：

```
#tar xzf FileSystem-Yaffs2.tar.gz
```

4.2、制作映像

在 Linux-3.0.1\filesystem\Yaffs2 文件系统制作工具 中有两个制作工具：
mkyaffs2image-nand2g 和 mkyaffs2image-nand256m

(1) mkyaffs2image-nand256m 制作出的映像，适用于 256M 的 nandflash 的平台
制作命令：

```
#./mkyaffs2image-nand256m FileSystem-Yaffs2 rootfs.yaffs2
```

(2) mkyaffs2image-nand2g 制作出的映像，适用于 1G, 2G 或者 4G 字节 nandflash 的平台
制作命令：

```
#./mkyaffs2image-nand2g FileSystem-Yaffs2 rootfs.yaffs2
```

最后在文件系统源码包同级目录下生成 rootfs.yaffs2 文件，此文件即是可下载到平台 nandflash 中的 yaffs2 文件系统映像。

假如用以上命令制作映像不成功，可能原因是制作工具权限不正确，需先修改下权限，修改权限的

命令：

```
#chmod 777 mkyaffs2image-nand256m
```

使用另外一个制作工具，修改权限的命令：

```
#chmod 777 mkyaffs2image-nand2g
```

注意：mkyaffs2image 可执行文件是使用 Linux3.0 源代码目录 yaffs2 文件夹下 utils 目录中的 mkyaffs2image.c 文件编译出来的，如果您有兴趣可以自己制作适合 256MB NandFlash 和 2G NandFlash 的 Yaffs2 工具。

4.3. 将 Qt 应用程序放入 yaffs2 文件系统的方法

这里我们以 Qt4.7.1 开发的简单界面应用程序 HelloWorld 及飞凌提供的 yaff2 文件系统为例，为您演示如何将自己开发的应用程序放入文件系统中。其中 Qt4.7.1 开发应用程序的方法，可以参考其他的资料，这里就不赘述了。

步骤 1，添加可执行文件到文件系统源代码目录中。

我们把 Qt 生成的可执行文件 HelloWorld 放到文件系统源代码/usr/bin 中，如下图所示：



步骤 2.添加运行可执行文件的脚本。

要运行此可执行文件需要配置下环境变量，我们把配置环境变量的语句都写到一个脚本里。在文件系统源代码的/bin 路径下新建一个文件，命名为 helloworld.sh，将配置环境变量和执行 HelloWorld 可执行文件的语句加进去，内容如下：

```
#!/bin/sh
export TSLIB_ROOT=/usr/local/tslib
export TSLIB_CALIBFILE=/etc/pointercal
export TSLIB_CONFFILE=/usr/local/tslib/etc/ts.conf
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_PLUGININDIR=/usr/local/tslib/lib/ts
export TSLIB_PLUGININDIR=$TSLIB_ROOT/lib/ts
export TSLIB_TSDEVICE=/dev/input/event2
export TSLIB_TSEVENTTYPE=H3600
export QTDIR=/opt/qt-4.7.1/
export
LD_LIBRARY_PATH=$QTDIR/plugins/QtOpenGL:$QTDIR/qt_plugins/imageformats:$QTDIR/lib:/root
/tslib/build/lib:$LD_LIBRARY_PATH
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/root/tslib/build/bin
if [ -c /dev/input/event2 ]; then
    export QWS_MOUSE_PROTO="Tslib:${TSLIB_TSDEVICE}"
    if [ -e /etc/pointercal -a ! -s /etc/pointercal ]; then
        rm /etc/pointercal
        /root/tslib/build/bin/ts_calibrate
    fi
else
    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
    >/etc/pointercal
fi
export QWS_KEYBOARD=TTY:/dev/tty1
FB_SIZE=$(cat /sys/class/graphics/fb0/virtual_size)
case "$FB_SIZE" in
    800,480)
        export QWS_DISPLAY="LinuxFb:mmWidth91:mmHeight53:1"
        ;;
    480,272)

```

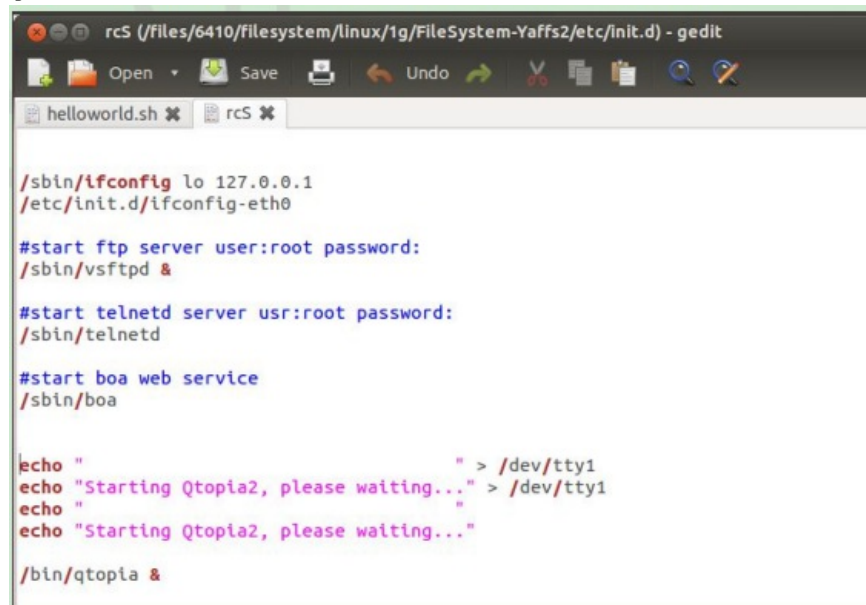


```
export QWS_DISPLAY="LinuxFb:mmWidth76:mmHeight44:1"
;;
*)
export QWS_DISPLAY="LinuxFb:mmWidth91:mmHeight53:1"
;;
esac
export HOME=/root/QtE4Home
chmod 777 /usr/bin/Helloworld
/usr/bin/Helloworld -qws &
```

添加完内容后保存。

步骤 3.将执行可执行文件的脚本设置成开机自启动。

为每次开机都自动运行此应用程序，则需要修改开机后运行的脚本 rcS 文件。打开文件系统源代码的/etc/init.d/rcS 文件，如下图所示：



```
rcS (/Files/6410/FileSystem/linux/1g/FileSystem-Yaffs2/etc/init.d) - gedit
helloworld.sh x rcS x

/sbin/ifconfig lo 127.0.0.1
/etc/init.d/ifconfig-eth0

#start ftp server user:root password:
/sbin/vsftpd &

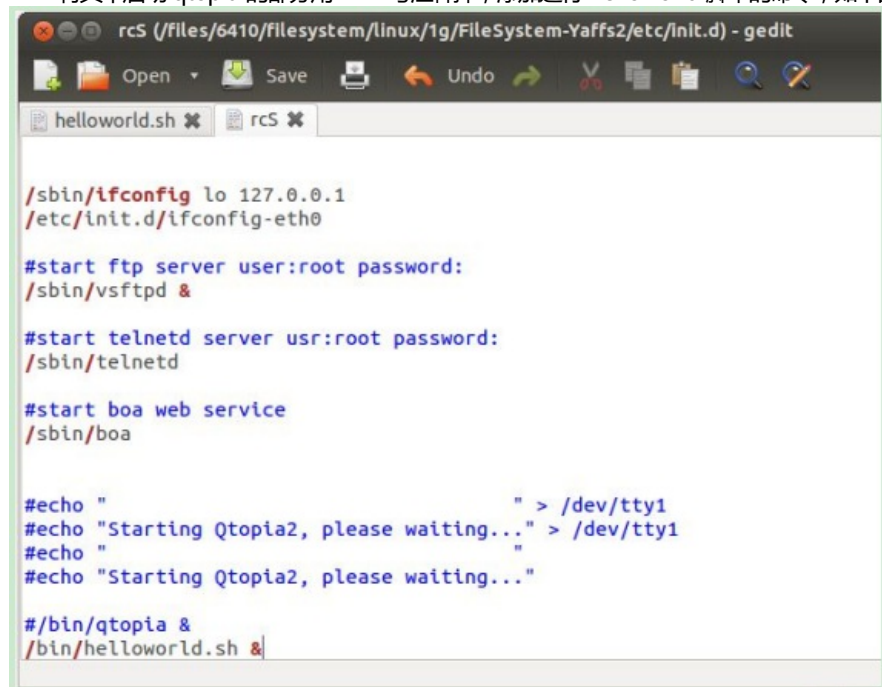
#start telnetd server usr:root password:
/sbin/telnetd

#start boa web service
/sbin/boa

echo " " > /dev/tty1
echo "Starting Qtopia2, please waiting..." > /dev/tty1
echo " "
echo "Starting Qtopia2, please waiting..."

/bin/qtopia &
```

将其中启动 qtopia 的部分用“#”号注释掉，添加运行 helloworld 脚本的命令，如下图所示：



```
rcS (/Files/6410/FileSystem/linux/1g/FileSystem-Yaffs2/etc/init.d) - gedit
helloworld.sh x rcS x

/sbin/ifconfig lo 127.0.0.1
/etc/init.d/ifconfig-eth0

#start ftp server user:root password:
/sbin/vsftpd &

#start telnetd server usr:root password:
/sbin/telnetd

#start boa web service
/sbin/boa

#echo " " > /dev/tty1
#echo "Starting Qtopia2, please waiting..." > /dev/tty1
#echo " "
#echo "Starting Qtopia2, please waiting..."

# /bin/qtopia &
/bin/helloworld.sh &
```

修改完后保存，退出。

以上则完成了文件系统源代码的修改。

步骤 4.制作包含可执行文件的 yaffs2 文件系统映像。

参考 9-2 制作映像一节制作 yaffs2 文件系统映像。

步骤 5.烧写文件系统映像，并开机看应用程序的运行。

将文件系统映像烧入开发板，启动开发板，会从界面上看到自己的应用程序 Helloworld 已经运行了。