编译 openwrt 系统需要安装的环境

yum install libtool autoconf automake gcc-multilib bison screen gcc g++ binutils patch bzip2 flex make gettext unzip libc6 git-core git build-essential libncurses5-dev zlib1g-dev gawk quilt asciidoc libz-dev

下载 OpenWrt 源码命令：

svn co svn://svn.openwrt.org/openwrt/trunk

更新种子列表：

./scripts/feeds update -a

./scripts/feeds install -a

配置系统：make menuconfig

编译源码：make -j 2 V=99

--------**软件安装与固件升级**----------

在 openwrt 下恢复固件（或使用 AUTOTTL 恢复固件）：

cd /tmp

wget http://ip/*.bin

mtd -r write *.bin firmware

更新 openwrt 固件(从低版本升级到高版本)：

cd /tmp

wget http://ip/*.bin

sysupgrade -i /tmp/*.bin

更新 opkg：

opkg update

安装 usb 支持：

opkg install kmod-fs-ext4

opkg install kmod-usb-storage

opkg install block-mount

--------**软件包管理、系统配置及分区修改等**----------

修改包配置（添加自己的程序包）：

trunt/package/

安装自己的程序包

make package/PGK_NAME install

例（helloworld 包）

1.将自己编写的源码包放到 openwrt/trunk/package 目录下，目录结构如下：
openwrt/trunk/package/helloworld/：其中包含 外层 Makefile 和 src/ 目录
openwrt/trunk/package/helloworld/src/：其中包含 内层 Makefile 和 源代码
2.运行：make package/PGK_NAME install 安装软件包

用 PUTTY 登陆进行 IP 地址配置

vi /etc/config/network

使 ip 配置生效：

/etc/init.d/network restart

系统应用的配置文件在：

/openwrt/trunk/package/base-file/files/etc/

openWRT 分区源码

target/linux/ar71xx/files/drivers/mtd/tplinkpart.c

修改 openwrt 的 flash 大小：

/yunho/trunk/tools/firmware-utils/src/mktplinkfw.c

修改系统 MAC，从 config 分区读取

修改文件 trunk/target/linux/ar71xx/files/arch/mips/ath79/mach-tl-wr741nd-v4.c

中的：u8 *mac = (u8 *) KSEG1ADDR(0x1f01fc00)为 u8 *mac = (u8 *) KSEG1ADDR(0x1ff20000);

openwrt 分区备份：

dd if=/dev/mtd5 of=/tmp/mtd5.bin （将 mtd5 分区拷贝到 tmp 目录下，文件名为：mtd5.bin）

开启无线及修改 SSID：

/openwrt/trunk/package/mac80211/files/lib/wifi/mac80211.

修改主机名

package/base-files/files/etc/config/system

配置网络服务：

/openwrt/trunk/package/network 和 /openwrt/trunk/package/base-files/files/etc/confit/network(核心文件)

LED、NTP、时区、路由器名的配置：

/openwrt/trunk/package/base-file/files/etc/config/system.

openwrt 系统 etc/目录下的配置文件及操作说明：

| www/ | luci web 的配置页面 |

| etc/banner/ | 命令行登录 openwrt 的欢迎信息 |

| etc/opkg.config | openwrt 的 opkg 更新源配置文件

| etc/profile | 系统环境变量

| etc/dnsmasq.conf | dns 配置文件

| etc/config/dhcp | dhcp 服务器配置文件

| etc/config/firewall | 防火墙配置文件

| etc/config/fstab | 文件系统挂载配置文件

| etc/config/luci | luci 界面配置文件

| etc/config/wireless | 无线配置文件
---

ttl 刷 art:
tftp 0x80000000 art.bin/
erase 0x9f7f0000 +0x10000 ---------8M 固件
cp.b 0x80000000 0x9f7f0000 0x10000
也可以用编程器刷，需要用 winhex 把 art 替换到编程器固件最后的 64K 部分，再刷。

安装 OpenWRT 工具链
make toolchain/install

分区读写控制
将分区的 mask_flags = MTD_WRITEABLE 为只读的；设置为 mask_flags = 0 为可读写。

gpio 口定义
trunk/target/linux/ar71xx/files/arch/mips/ath79/mach-tl-wr741nd-v4.c 中的
static struct gpio_led tl_wr741ndv4_leds_gpio[] __initdata 数组。

备份整个 openwrt 系统（从 u-boot 到文件系统）
Cat /dev/mtd0 /dev/mtd1 /dev/mtd2 /dev/mtd3 > /tmp/openwrtSYS

9331 相关

编译 9331 源码的命令

cd build

CPPFIAGS+=-D_FORTIFY_SOURCE=0 make BOARD_TYPE=ap121-2.6.31

9331 固件烧录

tftp 0x80060000 tuboot.bin

erase 0x9f000000 +0x40000

cp.b $fileaddr 0x9f000000 $filesize

tftp 0x80060000 ap121-2.6.31-squashfs

erase 0x9f050000 +$filesize

cp.b $fileaddr 0x9f050000 $filesize

tftp 0x80060000 vmlinux.lzma.uImage

erase 0x9f300000 +0xe0000

cp.b $fileaddr 0x9f300000 $filesize

9331 文件系统打包命令：

build/util/mksquashfs4.0 rootfs-ap121-2.6.31.optbuild

images/ap121-2.6.31/ap121-2.6.31-squashfs -noappend -b 16384 -all-root -pf

build/scripts/ap121-2.6.31/dev.txt

修改 9331 内核分区：

/9331/boot/u-boot/include/configs/ap121.h(修改内核分区、板子地址、PC 地址、MAC

地址)

9331MAC 地址定义：

boot/u-boot/cpu/mips/ar7240/ag7240.c

u-boot 启动时 挂载内核模块等操作：

/rootfs-ap121-2.6.31.optbuild/etc/rc.d/rcS 内核启动时启动 或

/linux/kernels/mips-linux-2.6.31/drivers 编译进内核

#例（挂载 art.ko）

 # 1.在 9331/rootfs-ap121-2.6.31.optbuild/lib/modules/2.6.31 目录下新建 **art/**目录将 **art.ko** 拷到 **art/**目录下

 # 2.修改 9331/rootfs-ap121-2.6.31.optbuild/etc/rc.d/rcS:在 USB 驱动之后，添加

insmod /lib/modules/2.6.31/art/art.ko

 # 3.执行 **build/util/mksquashfs4.0 rootfs-ap121-2.6.31.optbuild**

images/ap121-2.6.31/ap121-2.6.31-squashfs -noappend -b 16384 -all-root -pf

build/scripts/ap121-2.6.31/dev.txt 命令将文件系统打包

U-boot 部分启动的打印信息

9331/boot/u-boot/lib_mips/board.c

修改波特率、bootloader 延迟时间、I/O 缓冲区大小、命令行最大参数个数、DDR 内存修改

9331/boot/u-boot/include/configs/ar7240.h

u-boot 源代码的目录结构

1、board：----------中存放于开发板相关的配置文件，每一个开发板都以子文件夹的形式出现。

2、Commom：----------文件夹实现 u-boot 行下支持的命令，每一个命令对应一个文件。

3、cpu：----------中存放特定 cpu 架构相关的目录，每一款 cpu 架构都对应了一个子目录。

4、Doc：----------是文档目录，有 u-boot 非常完善的文档。

5、Drivers：----------中是 u-boot 支持的各种设备的驱动程序。

6、Fs：----------是支持的文件系统，其中最常用的是 JFFS2 文件系统。

7、Include：----------文件夹是 u-boot 使用的头文件，还有各种硬件平台支持的汇编文件，系统配置文件和文件系统支持的文件。

8、Net：----------是与网络协议相关的代码，bootp 协议、TFTP 协议、NFS 文件系统得实现。

9、Tooles：----------是生成 U-boot 的工具。

对 u-boot 的目录有了一些了解后，分析启动代码的过程就方便多了，其中比较重要的目录就是/board、/cpu、/drivers 和/include 目录，

常用命令

查看 CFE 传递给内核的命令行参数：

cat /proc/cmdline

备份整个系统

cat /dev/mtd0 /dev/mtd1 /dev/mtd2 /dev/mtd3 /dev/mtd4 /dev/mtd5 >

/tmp/16M

利用 tftp 传输文件命令：

tftp -r art.ko -g 192.168.1.10

启用程序 （以 mldonkey 为例）

etc/init.d/mldonkey start 或者 ./程序名

链接设备

ln -b /dev/ttyUSB0 /dev/ttyS0

安装自己独自编写的软件

make package/PGK_NAME install

单独编译内核模块安装包

make package/kernel/{compile,install} V=s

查看内核驱动

cat /proc/devieces

查询字符串所在文件

grep -r -n "字符" .

备份数据（将 mtd5 分区内容复制到 tmp/目录下）

dd if=/dev/mtd5 of=/tmp/mtd5

向分区写入数据(将 tmp 下的 mtd5 中内容写入到 proc 的第 6 个分区中)

mtd -r write /tmp/mtd5 /dev/mtd5

两台 linux 主机之间传输文件

scp yunho_cjx.tar.gz yunho@192.168.0.146:/home/yunho/yunho_cjx

定时执行程序 cron

cron 是 linux 下的定时执行工具，可以在无人干预的情况下运行作业。

cron 是 linux 内置服务，不会自动开启，需要用如下命令开启：

/sbin/service crond start

/sbin/service crond restart

/sbin/service crond stop

/sbin/service crond reload

如果想自动启动 crond 服务，需要在/etc/rc.d/rc.local 的末尾添加一句：

/sbin/service crond start

格式：* * * * * command

解析：前五个字符从左到右依次表示：分钟（0-59），小时（0-23），日期（1-31），月份（1-12），星期（0-6）

除了数字还有几个个特殊的符号就是"*"、"/"和"-"、","，*代表所有的取值范围内的数字，"/"代表每的意思,"*/5"表示每 5 个单位，"-"代表从某个数字到某个数字,","分开几个离散的数字。以下举几个例子说明问题:

每天早上 6 点

0 6 * * * echo "Good morning." >> /tmp/test.txt //注意单纯 echo，从屏幕上看不到任何输出，因为 cron 把任何输出都 email 到 root 的信箱了。

每两个小时

0 */2 * * * echo "Have a break now." >> /tmp/test.txt

晚上 11 点到早上 8 点之间每两个小时，早上八点

0 23-7/2，8 * * * echo "Have a good dream:)" >> /tmp/test.txt

每个月的 4 号和每个礼拜的礼拜一到礼拜三的早上 11 点

0 11 4 * 1-3 command line

1 月 1 日早上 4 点

0 4 1 1 * command line

每次编辑完某个用户的 cron 设置后，cron 自动在/var/spool/cron 下生成一个与此用户同名的文件，此用户的 cron 信息都记录在这个文件中，这个文件是不可以直接编辑的，只可以用 crontab -e 来编辑。cron 启动后每过一份钟读一次这个文件，检查是否要执行里面的命令。因此此文件修改后不需要重新启动 cron 服务。