Building Embedded Linux for Arm-Based Platform

我們這裡所使用的硬體以 Compaq PDA iPAQ 3600 series 爲例。iPAQ 3630 的 CPU 爲 Intel StrongARM SA-1110。

1. Preparation

爲了以後方便工作，我們建立以下的資料夾以及變數。

目錄結構

```
$ cd ~
$ mkdir project
$ cd project
$ mkdir build-tools images rootfs sysapps
```

project

- build-tools
- images
- rootfs
- sysapps
- tools

環境變數

```
$ vi setenv.sh
```

內容如下

```
export PROJECT=project
export PRJROOT=$HOME/${PROJECT}
export TARGET=arm-linux
export PREFIX=${PRJROOT}/tools
export TARGET_PREFIX=${PREFIX}/${TARGET}
export PATH=${PREFIX}/bin:${PATH}
```

2. Cross Toolchain

由於我們是使用 i386 Linux 的機器作爲開發平台(host)，爲了使得之後建置出來的系統可以在 arm 的平台(target)運作，因此我們需要使用 cross toolchain 來建置 embedded Linux，而不是使用一般在 i386 Linux 上的 toolchain。

```
$ cd ${PRJROOT}/build-tools
$ wget http://handhelds.org/download/toolchain/arm-linux-gcc-3.4.1.tar.bz2
$ tar xvjf arm-linux-gcc-3.4.1.tar.bz2
$ cd usr/local/arm/3.4.1
$ cp –a * ${PRJROOT}/tools
```

3. Kernel

下載及解壓縮 Linux kernel

```
$ cd ${PRJROOT}/build-tools
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.9.tar.bz2
$ tar xvjf linux-2.6.9.tar.bz2
$ cd linux-2.6.9
```

Linux kernel 2.6.9 中已經有內建 iPQA 3600 series 的設定檔，我們直接使用即可。

```
$ make h3600_defconfig
```

此預設值存在一些問題，爲了使 build kernel 順利進行，我們做了以下的變更。

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- menuconfig
```

取消

```
General setup    --->
Support CPU clock change (EXPERIMENTAL)
```

以及

```
ATA/ATAPI/MFM/RLL support    --->
Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
```

因爲我們是使用 flash rom 作爲儲存裝置，因此必須修改以下 source code。

```
$ vi drivers/mtd/maps/sa1100-flash.c
```

搜尋 h3xxx，將 static struct mtd_partition h3xxx_partitions[]改成如下所示

```
static struct mtd_partition h3xxx_partitions[] = {
    {
            .name           = "H3XXX boot firmware",
            .size           = 0x00040000,
            .offset         = 0,
            .mask_flags     = MTD_WRITEABLE,
    }, {
            .name           = "H3XXX root jffs2",
            .size           = MTDPART_SIZ_FULL,
            .offset         = 0x00040000,
    }
};
```

編譯核心映像

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- zImage
```

建立模組

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- modules
```

安裝核心

```
$ cp arch/arm/boot/zImage ${PRJROOT}/images/
$ cp System.map ${PRJROOT}/images/
$ cp vmlinux ${PRJROOT}/images/
```

```
$ cp .config ${PRJROOT}/images/
```

安裝模組

```
$ make ARCH=arm CROSS_COMPILE=arm-linux- \
> INSTALL_MOD_PATH=${PRJROOT}/images/modules modules_install
```

使用 Busybox 的 depmod.pl

```
$ cd ${PRJROOT}/sysapps/
$ wget http://busybox.net/downloads/busybox-1.00.tar.bz2
$ tar xvjf busybox-1.00.tar.bz2
$ cd busybox-1.00
$ cp examples/depmod.pl ${PREFIX}/bin
$ cd ${PRJROOT}/build-tools/linux-2.6.9
$ depmod.pl -k ./vmlinux -F ./System.map –b \
> ${PRJROOT}/images/modules/lib/modules > \
> ${PRJROOT}/images/modules/lib/modules/2.6.9/modules.dep
```

4. Root Filesystem

建立目錄結構

```
$ cd ${PRJROOT}/rootfs
$ mkdir bin dev etc lib proc sbin tmp usr var
$ chmod 1777 tmp
$ mkdir usr/bin usr/lib usr/sbin
$ mkdir var/lib var/lock var/log var/run var/tmp
$ chmod 1777 var/tmp
```

複製 libraries

```
$ cd ${TARGET_PREFIX}/lib
$ cp *-*.so ${PRJROOT}/rootfs/lib
$ cp -d *.so.[*0-9] ${PRJROOT}/rootfs/lib
$ cp libSegFault.so libmemusage.so libpcprocfile.so ${PRJROOT}/rootfs/lib
$ cp libSegFault.so libmemusage.so libpcprofile.so ${PRJROOT}/rootfs/lib
$ cd ${PRJROOT}/rootfs/lib
$ rm -rf libstdc++.so.6 libgcc_s.so.1
```

複製 kernel modules

```
$ cp -a ${PRJROOT}/images/modules/* ${PRJROOT}/rootfs
```

複製 kernel images

```
$ cd ${PRJROOT}/images
$ cp zImage ${PRJROOT}/rootfs/boot
```

建立 deivce files，需要有 root 的權限

```
$ cd ${PRJROOT}/rootfs/dev
```

```
$ su
```

下列表格是我們要建立的 device files

```
$ mknod -m 600 mem c 1 1
……
$ exit
```

Filename	Type	Major number	Minor number	Permission bits
mem	char	1	1	600
null	char	1	3	666
zero	char	1	5	666
random	char	1	8	644
tty0	char	4	0	600
tty1	char	4	1	600
ttySA0	char	204	5	600
tty	char	5	0	666
console	char	5	1	600
mtd0	char	90	0	664
mtd1	char	90	2	664
mtdblock0	block	31	0	664
mtdblock1	block	31	1	664

建置 Busybox，我們選擇完全安裝

```
$ cd ${PRJROOT}/sysapps/busybox-1.00
$ make allyesconfig
$ make TARGET_ARCH=arm CROSS=arm-linux- \
> PREFIX=${PRJROOT}/rootfs all install
```

最後，編輯以下三個檔案

/etc/inittab

```
$ vi {PRJROOT}/rootfs/etc/inittab
```

內容

```
::sysinit:/etc/init.d/rcS
::askfirst:/bin/ash
::ctrlaltdel:/sbin/reboot
```

```
::shutdown:/sbin/swapoff –a
::shutdown:/bin/umount -a –r
::restart:/sbin/init
```

/etc/fstab

```
$ vi {PRJROOT}/rootfs/etc/fstab
```

內容

```
/dev/root       /       jffs2       defaults 0 0
proc            /proc   proc        defaults 0 0
```

/etc/init.d/rcS

```
$ mkdir {PRJROOT}/rootfs/etc/init.d
$ vi {PRJROOT}/rootfs/etc/init.d/rcS
```

內容

```
#!/bin/sh
mount -n -o remount,rw /
mount /proc
```

5. Root Filesyatem Setup

製作 image 檔，我們使用 jffs2 檔案格式

```
$ cd ${PREFIX}/bin
$ wget ftp://sources.redhat.com/pub/jffs2/mkfs.jffs2
$ cd ${PRJROOT}
$ mkfs.jffs2 -r rootfs/ -o images/rootfs-jffs2.img
```