Android Porting Guideline for A1200

by androidezx@gmail.com 2008.08.05

1. Prologue

Now more and more Google android fans start to port Android to the real HW. The earliest successful android porting is achieved on Zaurus; later Android also run on OMAP based N810. And some developer ported Android onto PXA270 development board.

As we know, Motorola A1200 MING serial EZX PDA phones use Intel XScal-PXA270 application processor and Linux 2.4.20 kernel. After some investigation, it is fully possible to port android onto A1200 phone.

This document mainly describes the whole porting of android on A1200. For general Android knowledge, please refer to the relative documents.

2. Porting Android to A1200

Actually, porting Android is to port Google android kernel to the target (the real hardware) and let Android root file system can work with android kernel together on the target hardware.

Google provides android root file system with the form of binary and android kernel with the form of source in Android SDK. For Android root file system, we can easily extract it from Android SDK. But for Android kernel, we need to consider how to port it to our A1200.

There are two approaches for us to do porting android kernel to A1200. A) Porting all android driver specific changes into A1200. B) Upgrade A1200 kernel to 2.6.x then add android driver specific changes. It is obvious that we need to do more work if we use approach A) since Android kernel is based on 2.6.23 but A1200 kernel is 2.4.20.

If we choose approach B), we need to upgrade A1200 kernel to 2.6.x. Fortunately, there is one open source project called OpenEzx which can help us to upgrade A1200 kernel to 2.6.x easily. This document describes the detailed procedure to port Android on A1200 through approach B).

2.1 Android Kernel/Root File System

You can get the latest Android SDK kernel source (android-emulator-m5-rc14.tar.gz) from the below link http://code.google.com/p/android/downloads/list.

Android Root File System is provided by Google with the form of binary (EABI compatible) in SDK, you can download it from http://code.google.com/android/download_list.html.

In order to get Android specific patch against the specific 2.6.x Linux kernel, you need to diff Android SDK kernel with the official 2.6.23 kernel and make clear all Android driver specific changes. I have made one Android patch set for Linux 2.6.24 and provide it in porting package.

For Android Root File System, you may need to take some time to build it. You need to extract three images (ramdisk.img, system.img and userdata.img) provided in SDK to build up your root file system. You can refer to the relative part of Android on OMAP (http://elinux.org/Android_on_OMAP).

Since there is no existing workable root file system for A1200, I build up one new Android NFS Root File System with busybox toolkit and Android SDK by myself. As for NFS Root File System building up, I will not explain it more detail here, you can find many documents about it from the internet. I also provide Android NFS Root File System in porting package.

2.2 SW/HW environment requirement

After we prepare Android kernel and Root File System, now let us list all the software and hardware requirements:

2.1.1 Software requirement

Linux host:
You can use any popular Linux environment (like Redhat, Fedora, etc) as you like.

Tool-chain:
Since Android Root System provided in SDK was compiled with ARM EABI compatible compiler, we must use ARM EABI compatible tool-chain to compile our kernel. You can download ARM EABI compatible too-chain from the below link:
http://www.codesourcery.com/gnu_toolchains/arm/download.html

Note: arm-2007q3-51-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.tar is what you shall download, DON'T download other tar files.

Patch tool:
In order to upgrade our A1200 kernel, we need to add OpenEzx 2.6.24.x- patchset. OpenEzx use quilt tool to maintain patches. You can get this tool from
http://savannah.nongnu.org/projects/quilt.

USB Boot:
This tool developed by OpenEzx is used to boot up kernel via USB cable, you can get the latest version (0.2) from the link http://svn.openezx.org/trunk/src/host/boot_usb/.

Busybox 1.1.3:
Just like what I mentioned as above, I made Android NFS Root File System with busybox. During my porting, I use busybox 1.1.3 version. I attached busybox config file in package what I used for your reference.

2.1.2 Hardware requirement

We only need one A1200 phone and one standard USB cable for android porting.

Now it is time to start our detail porting procedure.

The below is my environment setup:
Linux Host: VMwareStation + TurboLinux 10
Tool-Chain: arm-none-linux-gnueabi- tool chain is installed in /home/arm-2007q3 directory.
Patch tool: quilt tool is installed in /home/quilt-0.46 directory
USB Boot tool: I compiled and put it under /home/testKernel/ezx_usb_boot

I created /home/android directory for kernel and /home/android_nfs directory for Android root file system.

2.3 A1200 Android Porting Procedure

1) Unzip Linux-2.6.24 kernel tar file (linux-2.6.24.tar.bz2) to /home/android/linux-2.6.24
   You can download linux-2.6.24.tar.bz2 from http://www.kernel.org/pub/linux/kernel/v2.6/.
2) Create patches directory under /home/android/linux-2.6.24 and copy all the files in OpenEzx patchset folder
3) Apply OpenEzx patches via cmd 'quilt push –a' under /home/android/linux-2.6.24.
4) Apply the following android patches via cmd 'patch –p2 <../ ***.patch' under /home/android/linxu-2.6.24 directory:
   - android-core.patch
   - android-config-nfs.patch
   - android-a1200-pcap.patch
   - android-a12000-16bpp.patch
   - android-framebuffer.patch
   - android-touchscreen.patch

Note: You can apply these patches without considering the order, but applying these patches must be done after openezx patch is applied successfully.

5) Modify the CROSS_COMPILE variable in Makefile.openezx of OpenEzx patchset as follows:
```
PHONES = a1200
CROSS_COMPILE ?= /home/arm-2007q3/bin/arm-none-linux-gnueabi-
```

Here I put tool chain under /home/arm-2007q3 directory. You can modify according to your tool chain directory.

6) Compile our android kernel via cmd 'make –f patches/Makefile.openezx under /home/android/linux-2.6.24 directory.

After about half an hour, the kernel and modules were compiled out successfully. You can see zImage-a1200 and modules-a1200.tar.gz under /home/android/linux-2.6.24 directory.

Now we can start Android via NFS:

A. Unzip android_nfs_root-m5-r14.bz2 in porting package to your specified directory.
   Note: This directory must be same with the specified directory in kernel cmd line parameter of defconfig-a1200. Here I use /home/android_nfs directory.

B. Unzip modules-a1200.tar.gz generated in the compile phase to the same directory with android_nfs_root-m5-r14.bz2. This tar was unzipped to /home/android_nfs/lib directory.
C. Insert USB cable into A1200 and let phone enter MBM flash mode (red screen)
D. Boot kernel via usb_boot tool under /home/testKernel via cmd './ezx_usb_boot zImage-a1200' After kernel is uploaded to A1200, you will see kernel is booting up from A1200 LCD (many kernel info is scrolling up).
E. After several seconds, telnet into A1200 phone from terminal of Linux host via cmd 'telnet telnet 192.168.1.2'
F. Run './init' execute after telnet into phone, you will see Android string
G. Run '/system/bin/runtime', after a while, you will see the exciting red eye cycle always running from left to right.

Again after about 2 minutes, you will see the familiar Android desktop. Then you can start Android application (Dialer, Map, Browser, etc) via touch screen. Certainly, if you modify kernel CMD_LINE parameter in defconfig-a1200 to let Android start from SD card, Android startup procedure can be speed up.

Currently, I modified Android keypad mapping to let Camera hard key as Make call key and VR hard key as End call key. You can click A1200 center select key of 5-direction navigation key to return to Android desktop.

Note: You can change kernel CMD_LINE parameter of defconfig-a1200 easily to generate MMC card version kernel, certainly you also need to modify /etc/init.rc file of Android root file system.

3. Current status & known issues

Now android can run many EZX phone like E680/A910/A1200. Here is the current status and existing issues:

Status:

- Android can boot up normally (Android string display, red eye cycle running, android desktop display)
- Touch screen can work on A1200/E680 basically
- Hard key (VR, Camera, Center select hard key) can work partially on A1200.
- Several major applications (Dialer, Map, Brower, Contacts) can be launched from UI.
- Audio does not work.

Issues:

- LCD display color is incorrect on A1200/A910 since A1200/A910 use 18 data pin physically for LCD panel, but Android root file system use 16BPP.
- Although Touch screen can work, still need calibration.
- Some other hard keys still don't work.

4. To do next

- Let BP MUX work on A1200
- Implement RIL (Radio Interface Layer) shared library to make GSM call
- Make Android audio work on A1200
- Make hard keys on A1200 work fully (if available)

- Touch screen calibration (if available)
- Fix A1200 LCD display color issue (not whether it can be fixed)

5. Some tips

- When building Android root file system by yourself, several critical dev nodes (/dev/binder, /dev/log/main, /dev/log/radio, /dev/log/event) must be created correctly.
- If your touch screen does not work, you should check whether touch screen driver is assigned to event1 via ‘cat /proc/bus/input/devices’ which Android application use /dev/input/event1 to get touch driver info.
- You can modify /system/usr/keylayout/qwerty.kl file to change Android key mapping.
- You can use strace to debug during your porting.

6. Reference

[1] OpenEZX homepage - http://www.openezx.org/

[2] Google Android homepage - http://code.google.com/android/

[3] Benno’s engineering Blog - http://benno.id.au/blog/

[4] Android on Zaurus - http://androidzaurus.seesaa.net/article/74237419.html

[5] Omegamoon Blog

http://www.omegamoon.com/blog/index.php?m=04&y=08&d=23&entry=entry080423-212550&category=1

[6]AndroidPortingOnRealTarget

http://wiki.kldp.org/wiki.php/AndroidPortingOnRealTarget?refresh=1

7. Deliverable

Except this porting guideline, I also attached some deliverables in porting packages, the below is deliverable list:

1) OpenEZX-patchset.zip – OpenEZX patchset for 2.6.24-x.
2) android-patches.zip – android specific patches including
   - android-core.patch
   - android-config-nfs.patch
   - android-a1200-pcap.patch
   - android-a12000-16bpp.patch
   - android-framebuffer.patch
   - android-touchscreen.patch
3) android-nfs-root-m5r14.bz2 – Android NFS root file system package.
4) android_nfs_kernel – Android test kernel
5) modules-a1200.tar.gz – Android test modules package
6) busybox.config – busybox config file for NFS root file system.

Thanks!

Firstly thanks OpenEzx project very much, it is this project which makes porting Android onto A1200 possible. Meanwhile I’d like to give great thanks to Zhang Alex who provided source of android-a1200-pcap.patch, as well as Cortez who provided source of android-framebuffer.patch.