VMWare で x86 版 Android

水野光男
日本 Android の会
組み込み WG

オープンソースの Android は x86 への移植が進められている。x86 用にビルドして、VMWare の仮想マシンやネットブックで試せるようになった。まだ移植されていないモジュールなどもあり、完全ではないが、Android の可能性を先取りして体験するには十分だ。

本稿では、Android を VMWare の仮想 x86 環境に移植する手順を説明する。ただし、Android のソースコードリポジトリは日々変化してるので、完全にこの通りの手順で移植できることを保証するものではない。

Android のビルド環境

Android のビルド方法は、1 月号で紹介した Android をソースコードからビルドした手順を参照して欲しい。ソースコードの入手先は次の URL だ。

```
http://source.android.com/download
```

ビルド環境の構築方法を簡単におさらいしておこう。Ubuntu 8.04LTS を使用する場合には、以下のようにパッケージのインストールと環境設定を行う。

```
 $ sudo apt-get install git-core gnupg sun-java6-jdk flex bison gperf libsdl-
dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev
zlib1g-dev valgrind
 $ mkdir ~/bin
 $ curl http://android.git.kernel.org/repo > ~/bin/repo
 $ chmod +x ~/bin/repo
 $ export PATH=~/bin:$PATH
```

ビルドのプロセスは多くのメモリを消費する。VMWare の Ubuntu 仮想マシンを使う場合は、割り当てるメモリを 1024MB 以上に変更しておこう。Ubuntu.vmx をエディタで開いて、memsize を変更する。

```
memsize = "1024"
```

cupcake の x86 用ビルド

公開されている Android のソースコードは、大きくわけて 2 つのブランチがある。1 つは製品として販売されてる HTC 製 G1 に搭載されているリリース版ブランチで、もう 1 つは cupcake と呼ばれる開発版ブランチだ。どちらのブランチでも x86 用にビルドができる。ビルド手順にも差異はない。

今回は開発版ブランチをビルドしてみよう。まず、ソースコードをダウンロードするために、cupcake ブランチを指定して、ローカルリポジトリを初期化する。

```
$ mkdir cupcake
$ cd cupcake
$ repo init -u git://android.git.kernel.org/platform/manifest.git -b cupcake
$ repo sync
```

このまま make コマンドでビルドすると、1 月号や 2 月号で紹介したように、SDK や Linux Zaurus で起動できる ARM 版の Android システムができる。x86 ビルドには、次のようなマニフェストファイルを作成する必要がある。

```
$ vi .repo/local_manifest
<manifest>
 <project name="platform/vendor/asus/eee_701" path="vendor/asus/eee_701"/>
</manifest>
```

Asus の EeePC 701 を指定している理由は、x86 用の Makefile や設定ファイルなどが EeePC 701 用として公開されているからだ。

```
http://android.git.kernel.org/?p=platform/vendor/asus/eee_701.git;a=summary
```

マニフェストファイルの変更を有効にするために、再度リポジトリの同期を行う。

```
$ repo sync
```

2009 年 4 月時点での cupcake 版では、バッテリードライバが存在しない場合、Android が再起動を繰り返し、起動ロゴから先に進まないという問題がある。下記 URL を参照し、パッチをあてる必要がある。

```
http://androidzaurus.seesaa.net/article/116995835.html
```

次のコマンドでビルドを開始する。

```
$ TARGET_ARCH=x86 TARGET_PRODUCT=eee_701 DISABLE_DEXPREOPT=true  \
  make installer_img
```

Ubuntu 8.10 でビルドする場合は、コンパイラのバージョン違いによる問題が発生する。make コマンドに次の設定を追加する。

```
 CC=gcc-4.2 CXX=g++-4.2
```

次のメッセージが出たらビルドは完了だ。

```
 Done with bootable installer image -
[ out/target/product/eee_701/installer.img ]-
```

カーネルのビルド

続いてカーネルをビルドする。インストールイメージは EeePC 701 用なので、VMWare の仮想マシンに必要なドライバが含まれていない。カーネルのソースコードは、ダウンロードしたソースコードの kernel ディレクトリにある。

```
 $ cd kernel
 $ make i386_defconfig
 $ make menuconfig
```

カーネルコンフィギュレーションで、ビルド環境のカーネルと混乱しないよう、Local version をつけておく。後に RAM ディスクを作成するので、一旦カーネルモジュールをインストールしなければならない。ビルド環境に影響を与えないための処置だ。

Android 関連では、以下のドライバが必要だ。

```
 General
  -  Enable the Anonymous Shared Memory Subsystem
 Driver
  -- Misc
   -- Binder IPC Driver
   -- Low Memory Killer
   -- High-speed in kernel logging driver
  -- Real Time Clock
   -- Android alarm driver
```

VMWare 関連では次の設定が必要になる。

```
 - Fusion MPT SPI
 - Network - Device - EISA, VLB, PCI
  -- AMD PCnet32 PCI
 - Graphics - Frame buffer
  -- VESA VGA
```

```
 - Console display driver
 -- Framebuffer console
```

1 つだけ、パッチの必要なソースコードがある。drivers/alarm.c でコンパイルエラーが発生する。このドライバは save_time_delta()という ARM 用の Kernel にしかない関数を使用している。
arch/arm/kernel/time.c に、その関数があるので、関数だけをコピーすればよい。

Android のソースコードは repo ツールと、git によって管理されている。ファイル編集をする前に、ローカルのブランチを作っておく方がよいだろう。

```
$ repo start mycake
```

こうしておけば、ソースコードに変更を加えても、リモートリポジトリと同期を取ったときに、ローカルでの変更もマージされるようになる。前述の time.c に変更を加えた場合、その変更を、ローカルリポジトリに追加し、コミットしておく。

```
$ git add drivers/rtc/alarm.c
$ git commit
```

カーネルとモジュールをビルドし、そのモジュールを一旦インストールし、RAM ディスクを作成する。

```
$ make bzImage && make modules
$ sudo make modules_install
$ mkinitramfs -o ~/initrd.img 2.6.27-android
```

VMWare 仮想ディスクへのインストール

ビルドした Android のイメージとカーネルをインストールするために、VMWare の仮想ディスクを作成する。

```
$ sudo apt-get install qemu
$ qemu-img create -f vmdk android.img 4G
```

このファイルを USB メモリスティックや NFS などを経由して、ビルド環境から VMWare 仮想マシンのホストとなる Windows PC にコピーする。

ここからは Windows PC での操作になる。また、作成した仮想ディスクにビルドした Android のイメージをインストールするために、Linux の仮想マシンイメージが必要だ。もしインストールしていないのであれば、
Ubuntu 8.04LTS の仮想マシンをダウンロードしよう。

```
http://www.ubuntulinux.jp/products/JA-Localized/vmware
```

まず、作成した仮想ディスク android.img を Ubuntu.vmx のあるフォルダに移動する。続いて、Ubuntu 仮想マシンで android.img をマウントできるように、VMWare の設定ファイルを書き換える。Ubuntu.vmx をエディタなどで開いて、下記を追加する。scsi:0:0 の記述の後がいいだろう。

```
scsi1.present = "TRUE"
scsi1.virtualDev = "lsilogic"
scsi1:0.present = "TRUE"
scsi1:0.fileName = "android.img"
scsi1:0.redo = ""
```

Android のビルドも Ubuntu 仮想マシンで行っていた場合は、一旦シャットダウンする。

```
$ sudo poweroff
```

つぎに、VMWare を起動する。これで、Ubuntu 仮想マシンから、新しく追加した android.img ファイルが仮想ディスクとして扱えるようになるので、Linux パーティションを作成し、ブートフラグをつける。VMWare の Ubuntu 仮想マシンにログインして、端末を開き、作業を続ける。

```
$ sudo fdisk /dev/sdb 改行
デバイスは正常な DOS 領域テーブルも、Sun, SGI や OSF ディスクラベルも
含んでいません
Building a new DOS disklabel with disk identifier 0xa5fa413e.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

警告: 領域テーブル 4 の不正なフラグ 0x0000 は w(書き込み)によって
正常になります

コマンド (m でヘルプ): n 改行
コマンドアクション
  e   拡張
  p   基本領域 (1-4)
p 改行
領域番号 (1-4): 1 改行
最初 シリンダ (1-522, default 1): 改行
Using default value 1
終点 シリンダ または +サイズ または +サイズ M または +サイズ K (1-522, default 522): 改行
Using default value 522

コマンド (m でヘルプ): a 改行
領域番号 (1-4): 1 改行

コマンド (m でヘルプ): p 改行
```

```
Disk /dev/sdb: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = シリンダ数 of 16065 * 512 = 8225280 bytes
Disk identifier: 0xa5fa413e

デバイス Boot      Start         End      Blocks   Id  System
/dev/sdb1   *           1         522     4192933+  83  Linux

コマンド (m でヘルプ): w改行
領域テーブルは交換されました！

ioctl() を呼び出して領域テーブルを再読込みします。
ディスクを同期させます。
```

作成したパーティションを ext3 でフォーマットしてから、マウントする。

```
$ sudo mkfs.ext3 /dev/sdb1
$ sudo mount /dev/sdb1 /mnt
```

ブートローダとして GRUB をインストールする。

```
$ sudo grub-install --root-directory=/mnt /dev/sdb
```

ここで、UUID を確認しておこう。

```
$ sudo vol_id /dev/sdb1 | grep "UUID="
ID_FS_UUID=<ディスクの UUID>
```

GRUB の設定ファイル menu.lst にブートの設定をする。

```
$ sudo vi /mnt/boot/grub/menu.lst

default         0
timeout         3

title   Android
root    (hd0,0)
kernel  /boot/bzImage root=UUID=<vol_id の結果> rw vga=788 init=/init
androidboot.hardware=eee_701
initrd  /boot/initrd.img
```

設定しているカーネルオプションを簡単に説明する。root は文字通りルートファイルシステムのデバイス名だ。UUID を使用しない場合は、root=/dev/sda1 にすればよいだろう。vga=788 はディスプレイドライバとして、vesafb フレームバッファドライバを使用し、SVGA（800x600）の解像度を使用する設定だ。VGA なら 785、XGA なら 791 を指定する。init=/init でカーネル起動後の初期化コマンドとして、Android の初期化コマンド/init を使用する設定をしている。最後の androidboot.hardware=eee_701 は、Android 独自のコマンドラインオプションで、設定したハードウェア名の初期化スクリプトが実行される。この場合、/init.eee_701.rc が実行される。

続いて、ビルドしたカーネルと RAM ディスクイメージをコピーする。

```
$ sudo cp ~/cupcake/kernel/arch/x86/boot/bzImage  ~/initrd.img  /mnt/boot
```

Android のシステムをコピーする。

```
$ cd cupcake/out/target/product/eee_701/
$ sudo su
# cp -a root/* /mnt
# cp -a system/* /mnt/system
# cp -a data/* /mnt/data
# cd /mnt/system/usr
# chmod -R 777 keychars
# chmod -R 777 keyclayout
```

keychars と keylayout のパーミッションを変更しないと、Android 実行時にキーボードレイアウトファイルの読み込みに失敗して、エラーが発生するからだ。

Android の初期化スクリプト/mnt/init.rc を、環境にあわせて書き換える。下記の mount コマンド各行の先頭に#を入れてコメントアウトすればよい。

```
# mount rootfs rootfs / ro remount
# mount ext3 /dev/block/sda6 /system
# mount ext3 /dev/block/sda6 /system ro remount
# mount ext3 /dev/block/sda8 /data
# mount ext3 /dev/block/sda5 /cache
```

これでインストールは終了だ。VMWare をシャットダウンする。

```
# poweroff
```

VMWare を作成した Android 仮想ディスクで起動するために、VMWare 設定ファイルを作成する。以下、再び Windows での作業になる。

Ubuntu.vmx をエディタで開いて、scsi0:0.fileName を android.img にし、scsi1:のエントリを削除し、

Android.vmx というファイル名で保存する。上書き保存しないように気をつけよう。

これですべての作業は完了だ。Android.vmx をダブルクリックして VMWare を起動する。しばらくすると、Android が自動的に起動する。

Android の操作

Android のキーボード操作について、簡単に触れておく。Enter キーで決定、Esc キーで戻る、Menu キーがメニューになっている。残念ながらマウスはまだあまり正しく動作していない。

Settings アプリケーションにある Locale and Text で Japanese を選択すると、メッセージやアプリケーション名などが日本語になる。ぜひ試してみていただきたい。

cupcake にはソフトウェアキーボードがインプリメントされている。しかし、残念ながら SVGA の画面サイズに対応していないようで、例外を起こして異常終了してしまう。上記、Locale and Text のオプションでソフトウェアキーボードを使用しないよう設定できる。cupcake は頻繁に更新されているので、近い将来に問題が修正されるかもしれない。

PC で USB メモリから起動

VMWare の仮想ディスクに行ったのと同じ手順で、USB メモリスティックにインストールすれば、PC で Android を起動することも可能だ。ただし、ネットワークなどのデバイスドライバは、それぞれの PC 用にビルドする必要がある。

Android をインストールをした USB メモリスティックを PC に差し込み、起動時に BIOS の設定で USB から起動するよう変更する。

GRUB の menu.lst ファイルで、カーネルのコマンドラインオプションに、root を指定するブロックデバイス名には、注意が必要だ。GRUB の root は(hd0, 0)だが、カーネルに指定する root は、カーネルがエニュメレーションを行う順序でデバイス名が変わる。カーネルは内蔵 HDD から順に、sda, sdb, …とデバイスを認識するので、内蔵 HDD の数によって、USB メモリスティックのデバイス名が変わる。内蔵 HDD が 1 つの場合は、USB メモリスティックは sdb になるだろう。同様に 2 つの場合は sdc になる。このような混乱を避けるために、USB メモリスティックから起動する場合は、UUID で root を指定した方がいいだろう。

おわりに

Android 開発版のブランチ cupcake を x86 で試してみる方法を記した。Android は携帯用フレームワークとして開発されているが、そのポテンシャルは携帯の枠にとどまらない。例えば、Instant-on Linux を搭載した PC が発売されているが、現在の X-Window に代わって、Android が使われることもあるのではないだろうか。