# VMWareで x86 版 Android

みずの みつお
mizmit1222@gmail.com
日本 Android の会
組み込み WG

Ubuntu の VMware 用仮想マシンを利用して、（比較的）お手軽に Android を起動する実験を紹介します。

# 目次

## 1) Ubuntu 仮想マシン

Ubuntu 仮想マシンは Ubuntu Japanese Team が配布している Ubuntu 8.04LTS を使用します。つぎのサイトからダウンロードします。

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

以下、この Ubuntu 上で作業します。ログインして、端末を開いておきます。

## 2) Android のソースコード入手

Android のカーネルのソースコードは、オープンソースのリポジトリの一部として公開されています。

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

今回は Android のシステムそのものをビルドする必要はないので、repo ツールをインストールして、ソースコードのダウンロードだけを行います。

```
$ mkdir ~/bin
$ curl http://android.git.kernel.org/repo >~/bin/repo
$ chmod a+x ~/bin/repo
$ export $PATH=~/bin:$PATH
$ mkdir cupcake && cd cupcake
$ repo init -u git://android.git.kernel.org/platform/manifest.git -b cupcake
$ repo sync
```

カーネルのソースコードは、cupcake/kernel にダウンロードされます。

## 3) カーネルのビルド環境

カーネルをビルドするためのパッケージをインストールします。

```
$ sudo apt-get update
$ sudo apt-get ugrade
$ sudo apt-get install build-essential kernel-package \
libncurses5 libncurses5-dev
```

## 4)　カーネルのコンフィギュレーション

カーネルのビルドオプションを設定します。VMWare で起動するための設定と、Android を起動するための設定の２つを考慮します。

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

設定項目のうち、下記の設定は必須項目です。

```
General setup --->
    (-android) Local version - append to kernel release
    [*] Enable the Anonymous Shared Memory Subsystem

Power management options --->
    [*] Wake lock

Device Drivers --->
    [*] Misc devices --->
          [ ] Android pmem allocator
          <*> Binder IPC Driver
          <*> Low memory killer

    [*] Fusion MPT device support --->
          <*> Fusion MPT ScsiHost drivers for SPI

    [*] Network device support --->
          [*] Ethernet (10 or 100Mbit) --->
                [*] EISA, VLB, PCI and on board controllers
                <*> AMD PCnet32 PCI support

Graphics support  --->
    <*> Support for frame buffer devices  --->
          [*] VESA VGA graphics support

<*> Real Time Clock  --->
    [*] Android alarm driver
```

簡単に理由を記します。

| | |
|---|---|
| Local version | initramfs を作成するためにモジュールインストールを行うので、Ubuntu のモジュールと区別するため |
| Anonymouos Shared Memory | 必須の Android 用ドライバ　（ashmem） |

| | |
|---|---|
| Wake lock | 必須の Android 用ドライバ　（wake lock) |
| Android pmem | Android の HAL ドライバだが、現状では G1 専用なので不要 |
| Binder IPC | 必須の Android 用ドライバ　（binder） |
| Low memory killer | メモリ不足時にプロセスを kill するドライバ |
| Fusion MPT | VMWare がエミュレートしている SCSI デバイスのドライバ |
| AMD Pcnet32 | VMWare がエミュレートしているネットワークデバイスのドライバ |
| VESA | フレームバッファドライバ |
| Android alarm | Android のアラームドライバ |

チューニングが好きな方は、これら以外のオプションはご自由に変更してください。

## 5)　カーネルのビルドとインストール

カーネルとモジュールをビルドして、initramfs を作り、インストールします。

```
$ make bzImage
$ make modules
$ sudo make modules_install
$ mkinitramfs -o initrd.img 2.6.27-android
$ sudo cp arch/x86/boot/bzImage /boot/bzImage-2.6.27-android
$ sudo cp initrd.img /boot/initrd.img-2.6.27-android
```

grub の設定をします。エディタで menu.lst を開きます。

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

つぎのエントリを追加します。

```
title     Android
root            (hd0,0)
kernel    /boot/bzImage-2.6.27-android root=UUID=... rw vga=788 init=/init
androidboot.hardware=eee_701
initrd    /boot/initrd.img-2.6.27-android
```

root=UUID=...の部分は他の grub エントリからコピーします。menu.lst ファイル先頭付近にある、hiddenmenu をコメントアウトしておくと便利かもしれません。

カーネルオプションについて、簡単に説明しておきます。

| | |
|---|---|
| vga=788 | VESA フレームバッファドライバを SVGA(800x600)で使用する。 |
| init=/init | カーネル起動後の初期化コマンドは/init を使用する。 |
| androidboot.hardware=eee_701 | 初期化スクリプト init.eee_701.rc を実行する。 |

## 6) Android の USB ブートイメージ

日経 ITPro で公開されている「話題の携帯向け OS「Android」を x86 パソコンで動かしてみよう」というすばらしい記事があります。Android をソースコードからビルドする手順が詳しく紹介されています。

```
http://itpro.nikkeibp.co.jp/article/COLUMN/20090219/325052
```

この方法でビルドされた USB からブートできるイメージファイルが公開されています。

```
http://www.miraclelinux.com/embedded/
```

今回は、このイメージを利用させていただいて、Android のインストールを行います。Miracle Linux さんのサイトから android-usb.img をダウンロードします。

```
$ cd ~
$ wget http://ftp.miraclelinux.com/pub/Android/android-usb.img
```

## 7) USB ブートイメージからのインストール

android-usb.img ファイルから Android のシステムを取り出すためにループバックマウントします。その方法は、つぎのブログで詳しく紹介されています。

```
http://d.hatena.ne.jp/androidzaurus/20090302/1235966325
```

ext3 ファイルシステムは 61 セクタから始まっていて、 1 セクタが 512 バイトなので、31232 バイトのオフセットをつけてマウントします。

```
$ sudo losetup -o 31232 /dev/loop0 android-usb.img
$ sudo mount /dev/loop0 /mnt
```

ここから、必要なファイルをルートにコピーして、インストールします。

```
$ sudo mkdir /system /data
$ sudo cp -a /mnt/system/* /system
$ sudo cp /mnt/init* /
$ sudo cp /mnt/default.prop /
$ sudo mv /mnt/system/etc/hosts /tmp/hosts.android
$ sudo cp -a /mnt/system/etc/* /etc
$ sudo mv /tmp/hosts.android /mnt/system/etc/hosts
```

簡単にファイルの説明をします。

| | |
|---|---|
| /system | Android システムのほぼすべて |
| /data | 空のまま |
| /init* | 初期化コマンドとスクリプト |
| /default.prop | デフォルトプロパティ |
| /system/etc | 設定ファイル |

Ubuntu では、これらのディレクトリ名、ファイル名はルートに存在しませんので、名前の衝突を気にしなくてもすみます。

例外は/etc です。Android の/etc は/system/etc へのシンボリックリンクになります。しかし、/system/etc にある設定ファイルは、Ubuntu の/etc にある設定ファイルと、hosts ファイルをのぞいて名前の衝突がありません。そこで、/system/etc 以下の hosts 以外のファイルを/etc にコピーしてしまいます。

/data ディレクトリが空のままでいい理由は、Android は/data ディレクトリを初回起動時に自動的に生成するからです。Android はフラッシュメモリを搭載した携帯電話を想定したフレームワークです。フラッシュメモリが壊れたとき、あるいは工場出荷時の状態に戻すときに、/data ディレクトリを消してしまうだけで済むように実装されています。Android がマウントするシステムパーティションのうち、ルートや/system はリードオンリーでマウントされるので、破損する恐れはほとんどありません。

一方、/data はリード／ライトでマウントされるので、突然電池が切れるなどのトラブルで、ファイルシステムが壊れてしまう可能性があります。ですので、このような実装は理にかなっていると言えます。

インストールはこれで終了です。Ubuntu 仮想マシンをリブートします。

```
$ sudo reboot
```

## 8)　Android の起動

VMWare が再起動し、「grub loading...」のメッセージが表示されたら、[ESC]キーを押して、ブートメニュー画面を表示させます。カーソルキーで「Android」を選択して起動します。

Android の UI では、[ESC]キーがバックボタン、[Menu]キーがメニューボタンに割り当てられています。

残念ながら[END]キーの長押しによる電源オフには対応していないようです。VMWare の[Trouble shoot]メニューから、[Power off and exit]を選択して終了します。

## 9)　できないこと

サウンド。音は鳴りません。

Map, Gmail など。オープンソース版に含まれていません。

ソフトウェアキーボード。例外終了します。

## 10)　まとめ

Android 用カーネルの設定

　Android が起動できるカーネルのビルドオプションを確認しました。。

Android システムのディレクトリ構造

　Android システムのディレクトリ構造は、既存の Linux ディストリビューションと共存できます。

x86 で Android

　x86 で Android が動くとこを確認しました。