解説

Linux と USB bootable Android †

市村 匠 *1・木暮 祐一 *2

1. はじめに

最近、AndroidスマートフォンやタブレットPCなど、モバイル情報端末機器に搭載されているAndroidは、Googleがモバイル向けプラットフォームとして開発したオープンソースであり、Linuxのカーネル、ミドルウェア、アプリケーションから構成されるソフトウェアパッケージとなっている。オープンソースということで、搭載ソフトウェアのライセンス料を圧縮したい端末メーカーからも注目され、Android搭載スマートフォンが急激にシェアを拡大している。米国の調査会社Canalysによれば2010年第4四半期には全世界のスマートフォン市場において、NOKIAが提供してきたSymbian OSを上回り首位に立った[1]。今後もモバイル情報端末機器向けプラットフォームのスタンダードとして定着していくものと考えられる。特にアプリケーションについては、Android Marketと呼ばれるダウンロードサイトから優れた機能をもつソフトウェアをダウンロードすることができる。2011年3月におけるアプリ数は、iPhone OS向けの35万本に対し、Android向けは25万本まで迫っており、年内にはiPhone OS向けを追い抜き、スマートフォン向けアプリケーション数でもトップになるとされている。このように、モバイル向けプラットフォームとしてAndroidはより重要なものとなっていく。

Androidは、ハードウェアを操作するLinuxカーネル、標準ライブラリやAndroidランタイム、アプリケーションフレームワーク、アプリケーションソフトウェアから構成される。Androidランタイムには、Dalvik VMと呼ばれる仮想マシンを含んでいる。これはJava仮想マシンと考えられることもあるが、Java仮想マシンよりメモリが少ない環境に対して最適化されており、少ないメモリでも動作する。これによって、

標準Cライブラリが再実装されるようになり、Java仮想マシンと異なる一つの根拠となっている。またLinuxカーネル、標準ライブラリ、AndroidランタイムはC言語やC++言語で開発されており、アプリケーションはDalvik VM上で動作するJavaのサブセットとAndroid固有の環境で開発されている。したがって、Javaでプログラミングをしたことがある人であれば、Androidアプリケーションの開発は容易であろう。

本解説記事を2回に渡って連載する。まず今回は「Linux とUSB bootable Android」として、LinuxとAndroidの歴史を述べるとともに、Android PCのインストール、Android開発とエミュレータについて説明する。携帯アプリケーション開発を授業に取り入れている大学もあり、アプリケーション開発を考えている研究者や学生の皆さんにとって、有用な資料になるだろう。次回は「Android 端末におけるソフトコンピューティング・インタフェース」として、Android端末を用いたソフトコンピューティングアプリケーションについて解説する予定である。

2. Linux 上の Android

図1の概略図[2]に示すように、Androidでは最下層において、ハードウェアを動作させるためのドライバは、Linuxに依存している。ディスプレイ、カメラ、WiFi、SDカードへのアクセスなどは、Linuxカーネルに含まれるドライバにより操作される。Linuxも最

図1 Android アーキテクチャ概略図

† USB bootable Android on Linux

Takumi ICHIMURA and Yuichi KOGURE

*1 県立広島大学 経営情報学部 経営情報学科

Faculty of Management and Information Systems, Prefectural University of Hiroshima

*2 武蔵野学院大学 国際コミュニケーション学部 国際コミュニケーション学科

Faculty of International Communication, Musashino Gakuin University

表1 Android と Linux のバージョン

バージョン	コードネーム	カーネル
1.5	Cupcake	2.6.28
1.6	Donut	2.6.29
2.0/2.1	Eclair	2.6.29
2.2	Froyo (Frozen Yogurt)	2.6.32
2.3	Gingerbread	2.6.35
3.0	Honeycomb	2.6.36

近は落ち着いてきたものの、未だに開発が行われており、様々なバージョンのカーネルがある[3]。当然、Android もそのバージョンによって、用いる Linux カーネルのバージョンが異なっている。

Androidのバージョンは、コードネームがつけられているが、その頭文字がアルファベット順に名づけられていることはご存知だろうか。表1を見ると、甘いお菓子の名前の一覧になっている(開発者はよほど甘いものが好きなのだろう)。Androidスマートフォンの購入者が増えたのは、Android1.6からであろうか。このころから、Google Latitude(位置情報共有)[4]が使えるようになり、GPS携帯の新しい使い方が模索され始めてきている。Android2.0では、Googleマップ[5]が改良され、携帯電話のアプリに対応できるよう、Java CDCのAPI(Application Program Interface)が使えるようになった。Android2.2ではFlashにも対応するようになった。これ以外にも、Gmailやカレンダーとの連携で、まさにAndroidスマートフォンは小型化されたPCになっていった。さらにAndroid3.0はTablet PCでの利用を想定した大型ディスプレイに対応した新しいUIを採用しており、“ケータイ”からスマートフォンへの進化が着々と行われている。

3. Android PC

Androidが動作するのは、ARMプロセッサ(Advanced RISC Machines)のような携帯電話で動作するチップだけではない。x86アーキテクチャで動作するように開発を進めているProjectがある[6]。これは、Android-x86と呼ばれ、x86マシン(通常のPC)などで動作する。つまり Windows や Linux だけでなく、新たな OS として Android を用いることも可能である。インストールのために、まずWebサイト[6]から自分の環境にあった ISO イメージ(CDフォーマットの規格)をダウンロードする。通常は“android-x86-2.2-generic.iso”のイメージであろう。

表2 Linux のブートの仕組み

Step 1	PCの電源を入れる。
Step 2	BIOSを起動する。
Step 3	MBRを読み込む。
Step 4	ブートローダを起動する。
Step 5	カーネルを読み込む。
Step 6	ハードウェアのチェック、ドライバの読み込み、初期化を行う。
Step 7	ルートシステムをマウントする。
Step 8	init(Linuxの最初のプロセス)を起動し、他のプロセスを実行する。
Step 9	init レベルが5であれば、グラフィカルログイン画面が表示される。

図2 起動時のセレクト

このイメージをCD-Rに焼けば他のLinuxと同様にOSとしてインストールできる。CD-Rを挿入して起動すればよい。Linuxは、表2で見られるような手順でOSを起動するが、これにより1)PCの電源投入、2)BIOS起動、3)MBR(Master Boot Record)を読み、4)ブートローダ(GRUBなど)を起動する。MBRはハードディスクドライブの先頭セクタであり、GRUBはOSを起動するために最初に読み込まれるプログラムである。

ここまで進むと、図2が表示される。どのAndroidを読み込むのかを選択できるが、最初は一番上の“Live CD-Run Android without installation”を選んで、インストールせずにAndroidを起動させるとよい。ここでHDPIはHigh Density Per Inchを、MDPIはMedium DPIを示している。Android端末の画面解像度は、機種によって異なっており、開発時のデザイン目

安として、HDPI が240dpi, MDPI が160dpiを想定している。たとえば、私がメインで使用している機種は LYNX 3D, GALAXY TABであるが、これらはともに 240dpi/HDPI である。なお Linux は表 2 の Step 9 まで実行され、initレベルが3の状態、つまり、グラフィカル・ログイン画面ではなく、コンソールでシステム管理者(root)でログインした状態となる。この状態からAndroid が起動するようになっているのが、Android-x86である。

選択すると、図3のように、画面の中央にAndroidのロゴが表示される。正常に起動すると、図4のようなデスクトップが表示される。ここで、アプリケーションのボタン(16個の四角形の部分)をマウスでクリックすると、図5のようなメニューが表示され、設定を行うことが可能になる。

設定アプリケーションを起動すると、図6のような設定画面が表示されるので、「Ethernet Configuration」で、ネットワークの設定を行えば LAN に接続できる。たとえば、ブラウザで日本知能情報ファジィ学会のホームページを見ると、図7のように閲覧できる。

さて、この ISO イメージは CD-R に焼かれ、CD-R から Linux をロードし、その上で Android を起動している。この過程は、基本的には KNOPPIX[7] で PC を起動する過程と同じである。KNOPPIXは、1 CD bootable Linux として知られ、以前、USB bootable KNOPPIX を開発していたチームがあった。ここでは少ないメモリ容量しかない PC で、BIOS で USB からの起動が可能で

図5 Android アプリケーション

図6 Android 設定メニュー

図3 起動時中の Android ロゴ

図4 Android デスクトップ

図7 Android ブラウザ

図 8 Windows 上の Unetbootin

あるなら、USBメモリに保存されているLinuxを起動させるというものであった。CD Linuxよりも起動までの時間が速くなるという利点があった。

最近、Androidが注目を浴び、ミラクル・リナックス社と日経BP社日経Linux紙の共同企画で、USB bootable Androidが開発されている[8]。Linux上で、USBのブートイメージをUSBメモリに書き込むため、次のようにLinuxのコマンドddを用いている。つまり[8]のサイトでは、USBメモリ用Androidを構築し、イメージとして提供しているというものである。

```
# dd id=android-usb.img of=/dev/sdb
```

ここで、android-usb.imgはダウンロードしたイメージファイル名、/dev/sdbはUSBメモリのデバイス名である。

一方、ISOイメージをCD-Rに焼かずに、USBメモリに変換するツールがフリーで存在している[10]。UNetbootinは、Linux、Windows、MacのOSに対応しており、LinuxのISOイメージをそのままUSBメモリに変換することができる。図8は、Windows上で動作させたUnetbootinである。非常に容易にUSB bootableイメージを構築することができる。これによって、インストールしなくても、Android PCを利用する環境が整った。ただし、開発は4節で述べる環境で開発するのが望ましく、このAndroid PCは実装したソフトウェアを利用するために用いるのがよいだろう。

4. Androidアプリケーション開発環境の構築

WindowsでのAndroidアプリケーション開発は、書籍やGoogle検索でもたくさん紹介されている。こ

では、Linux上でのAndroidアプリケーションの開発について説明する。開発にはJDK6、Android SDK、Eclipse、Pleiades(Eclipseの日本語化Plugin)、EclipseのAndroid Pluginが必要となる。ここでは、Fedoraを例にとって説明する。

4.1 JDK

AndroidアプリケーションはJavaで作成するので、まずOracle社からJava SE Development Kit(JDK)をインストールする。本稿執筆段階ではJDK6が最新である。Platformを選択し、JDKをダウンロードする。

```
$ chmod +x jdk-6u2x-linux-xxx.bin
# ./jdk-6u2x-linux-xxx.bin
```

ここで、'6u2x'はバージョンを、'xxx'は32bitか64bitを示している。

Linuxの場合、ログイン時に環境変数設定ファイルを読み込むので、~/.bashprofileファイルで、環境変数を設定する。

```
export JAVA_HOME=/usr/java/jdk1.6.0_xx
export PATH=$PATH:$JAVA_HOME
export CLASSPATH=.:$JAVA_HOME
```

4.2 Android SDK

Android SDKをインストールする。

1. Webサイト[9]から、パッケージを選択し、Android SDKをダウンロードする(本稿執筆の時点では、android-sdk_r10-linux_x86.tgz)。
2. 適当なディレクトリに展開し、展開したディレクトリのtoolsディレクトリにあるandroidのファイルモードを実行できるように変更し、androidへのPATHを~/.bash_profileファイルで設定する。

```
$ chmod +x
~/android-sdk-linux_86/tools/android
```

PATHが設定されていれば、

```
$ android
```

とすると起動する。ちなみに、Android SDKのみでもエミュレータを実行することができる。しかしながら、アプリケーション開発には、統合環境であるEclipseを利用の方が便利である。

インストールしたディレクトリをPATH環境変数に追加するか、後述のEclipseでandroidへのパスを設定する必要がある。

4.3 Eclipse

LinuxではRPMなどソフトウェアのパッケージを管理する方法があり、Fedoraではyumを用いてインストールすると、必要な環境が整っていないくても、依存関係のあるソフトウェアパッケージも同時にインストールされる。Eclipseをyumでインストールする方法もあるが、Androidプラグインをインストールするときに、不具合が生じることがあるので、オリジナルのパッケージを[11]からダウンロードする。ここでは、“Eclipse IDE for Java Developers”をダウンロードし、/usr/localにインストールする。

```
# cd /usr/local
# tar zxvf
    eclipse-java-helios-SR1-linux-gtk.tar.gz
# chmod a+x eclipse/eclipse
```

インストールしたEclipseは日本語化されていないので、日本語化したい場合はPleiades[12]からダウンロードし、インストールする。Pleiadesは、Javaアプリケーションを実行時に翻訳するためのツールである。ダウンロードした圧縮ファイルをEclipseをインストールしたディレクトリ(ここでは、“/usr/local/eclipse/”)に展開し、同じファイルは全て上書きする。

Eclipseを起動するときは、/usr/local/eclipse/eclipseで起動する。

4.4 Android プラグイン

Eclipse上でAndroidアプリケーションを開発するためには、“Android Development Tools(ADT)”をインストールしなければならない。Eclipseの「新規ソフト

図10 ADTのリスト

ウェアのインストール」を使ったダウンロード方法を説明する。Eclipseを起動した後、「ヘルプ」→「新規ソフトウェアのインストール」で、図9のように、「使用可能なソフトウェア・サイト」からADTを選択するか、自分でロケーションに“https://dl-ssl.google.com/android/eclipse/”を入力する。すると自動でサイトにアクセスし、図10のように、必要なパッケージリストを取得し、使用許諾を承認すると、インストールが始まる。Eclipseを再起動すると、Androidの開発ができる環境が構築されているだろう。なお、Google Map APIなどを用いる場合は、このプラグインを指定することで、ダウンロードすることができる。

4.5 Android のアプリケーションの開発

Eclipseを使ってアプリケーションを作るためには、次の手順が必要となる。

図9 ADTのインストール

図11 新規プロジェクト作成

図 12 プロジェクトなどの設定

1. Eclipseを起動し、図11のように、「ファイル」→「新規」→「新しいプロジェクト」を選択する。
2. 図12のように、「Android Project」を選択する。プロジェクト名、ターゲット(表1)などを入力する必要がある。これは、通常のJavaで開発する手順と変わらない。
3. 「実行」ボタンを押すと、エミュレータが起動する。なお、Android ケータイとPCをUSBケーブルで接続し、エミュレータを起動すると、実機にアプリケーションが転送され、ケータイでの動作確認も行える。

5. Widgetの作成

Android SDKには、Widgetを作成するためのAppWidget frameworkがある。AppWidgetを作成するためには、次の項目が必要である。

1. AppWidget レイアウトの作成
2. AppWidget 設定ファイルの作成
3. AppWidget 処理の記述
4. AndroidManifest.xml への登録

HelloWorldというプロジェクトを作ると、レイアウトは“HelloWorld/res/layout/main.xml”(図13)、設定ファイルは“HelloWorld/res/xml/appwidget.xml”(図14)、マニフェストは“HelloWorld/AndroidManifest.xml”(図15)のように保存される。マニフェストはアプリケーションに関する情報をAndroid OS側に提供する役割があり、端末の回転時の操作、アドレス帳へのアクセスなどの権限に対する制限を指定することができる。AppWidgetの処理は、AppWidget Provider クラスから継承される処理をオーバーライド

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://
    schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

図 13 main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android
    .com/apk/res/android">
</appwidget-provider>
```

図 14 appwidget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas
    .android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0">

    <application android:icon="@drawable
        /icon" android:label="@string/
        app_name">
        <activity android:name=".HelloWorld"
            android:label="@string/
            app_name">
            <intent-filter>
                <action android:name="android
                    .intent.action.MAIN" />
                <category android:name="android
                    .intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

図 15 AndroidManifest.xml

するための処理で、AppWidget の呼び出し、更新、削除などのイベントに対する処理をJavaで書く。

6. 開発したアプリのダウンロード

開発したアプリケーションは、binディレクトリに拡張子がapkのファイル名で保存されている。このapkはJavaのjarファイルと同じように、zip形式のアーカイブとなっている。自分で開発したアプリケーションはAndroidマーケットによる承認を受けていないので、サードパーティに分類されてしまう。Androidでは、「設定」→「アプリケーション」→「提供元不明のアプリ」を受け入れるように設定すれば、自分で開発したアプリケーションも使うことができる。

Android PCでは、USBや直接ダウンロードして、ファイルをインストールすることができる。一方、Android エミュレータからlocalhost(127.0.0.1)には接続できない。これは、エミュレータで示すlocalhostがAndroid エミュレータ自体に接続してしまうからであろう。エミュレータで動作しているAndroidヘッファイルダウンロードする場合、EthernetのIPアドレスを「10.0.2.2」としてダウンロードする。図16は、エミュレータでダウンロードした時に表示される画面で、該当するアプリケーションをクリックすればインストールが始まる。

図 16 ダウンロード履歴

7. まとめ

本稿では、USB bootable Androidにより、DOS/V PCでも動作するAndroidを紹介した。また、Linux上で開発環境を構築し、簡単なアプリケーションを開発した。Android端末では、タッチパネルの操作性が重要視されるところである。Widgetはその環境を提供するためには不可欠と言える。

この記事を書く前に、すでにAndroid端末を用いて研究をされている方と話す機会があった。同じ開発者の立場から、Google APIが使えるため、容易に開発できるということがあった。また、通信が3GだけでなくWiFiも使えることで、“ケータイ”ネットワークの制限にとらわれずに、自由に利用できる場所に魅力を感じるということであった。研究のために、実際にAndroidアプリケーションを開発し、自分のケータイで動作させているが、まるで操作性のよいMini PCを持っているような錯覚にとらわれる。

しかしながら、メーカーによってカメラやSDカードの取扱いが異なっており、これらが開発上の問題点となっている。例えば、大容量のデータを取り扱うためにSDカードを用いるが、SDカードのマウント場所の違いなど、機種によって異なっている（機種ごとの違いを吸収するライブラリを使っても、問題が起こる）。これらは、Linuxのカーネル上で、デバイスをどのように制御するかという仕組みであろう。Linuxでも多様なディストリビューションがあるように、Androidのバージョンにも違いがあって、これらのバリエーションを考えると、実機でのチェックも必要であろう。

さて、本解説は、次回とあわせて2回の連載を予定している。次回は、Androidをどのように研究に用いるか、その具体的な例を踏まえて説明する予定である。また、2011年4月1日からスタートした学会SNSにおいて、研究グループ「Androidケータイを用いたソフトコンピューティング」を開設した。関心のある方は参加されたい。

謝辞

本原稿を執筆するにあたり、県立広島大学経営情報学科4年鎌田真君に動作環境の確認、及びAndroidアプリケーションの開発をお願いした。ここに謝意を表す。

参 考 文 献

- [1] Google's Android becomes the world's leading smart phone platform, <http://www.canalys.com/pr/2011/r2011013.html>, 2011 年 4 月 3 日.
 - [2] Android アーキテクチャ, <http://developer.android.com/images/system-architecture.jpg>, 2011 年 3 月 28 日.
 - [3] Linux カーネルマップ, http://www.makelinux.net/kernel_map, 2011 年 3 月 28 日.
 - [4] Google Latitude, http://www.google.com/intl/ja_jp/latitude/intro.html, 2011 年 3 月 28 日.
 - [5] Google Maps, <http://maps.google.co.jp/>, 2011 年 3 月 28 日.
 - [6] Android X86 Project, Android-x86, <http://www.android-x86.org/>, 2011 年 3 月 28 日.
 - [7] KNOPPIX, <http://www.knopper.net/>, 2011 年 3 月 28 日.
 - [8] Embedded MIRACLE, Android USBブートイメージダウンロードサイト, http://www.miraclelinux.com/embedded/android_dl.html, 2011 年 3 月 28 日.
 - [9] Android Develops サイト, <http://developer.android.com/intl/ja/index.html>, 2011 年 3 月 28 日.
 - [10] UNetbootin, bootable Live USB, <http://unetbootin.sourceforge.net/>, 2011 年 3 月 28 日.
 - [11] Eclipse, <http://www.eclipse.org/downloads/>, 2011 年 3 月 28 日.
 - [12] Pleiades, <http://mergedoc.sourceforge.jp/>, 2011 年 3 月 28 日.
- (2011 年 4 月 20 日 受付)

[問い合わせ先]

〒734-8558 広島市南区宇品東一丁目1番71号
 県立広島大学経営情報学部経営情報学科
 市村 匠
 TEL: 082-251-9571
 E-mail: ichimura@pu-hiroshima.ac.jp

著 者 紹 介

いちむら たくみ
市村 匠 [正会員]

1997 年桐蔭横浜大学大学院工学研究科博士後期課程修了。博士(工学)。広島市立大学情報科学部を経て2010年県立広島大学経営情報学部准教授。ニューラルネットワーク、進化計算、自己組織化マップ、人工免疫システムや、医療診断支援システム、感情インタラクションシステムの研究開発に従事。SOFT、IEEEなどの会員。IEEE SMC Hiroshima Chapter Chair (2011-)。趣味はAndroidアプリ開発などソフトウェア開発全般。

きくち ゆういち
木暮 祐一 [非会員]

1990 年杏林大学保健学部卒。株式会社アスキー『携帯24』編集長、株式会社ケイ・ラボラトリー(現、KLab)広報宣伝部マネージャーなどを経て、2003年戸板女子短期大学非常勤講師。2007年徳島大学大学院工学研究科修了。博士(工学)。2009年武蔵野学院大学国際コミュニケーション学部准教授。モバイル端末の遠隔医療分野、教育分野への活用に関わる研究に従事。モバイル学会理事、日本遠隔医療学会、日本教育工学会、ヒューマンインターフェイス学会など各会員。