ソースコードを活用しよう

マイSDKを作ってみる

自己紹介

・名前　　：robo（兼高理恵）

・お仕事　：Java技術者

・好きな物：モバイル端末

マイSDKを作ってみる

何のために、ソースコードから
マイSDKを作るの？

・ソースコードに慣れよう

・makeは友達怖くない

・いろいろお得があるから

マイSDKを作る手順は？

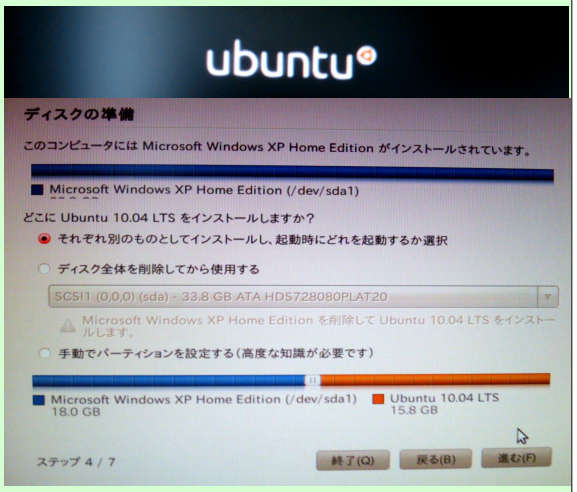
- Ubuntu 10.04 インストール
- 環境設定 （ソース取得＆ビルド）
- ソースコード取得
- SDKビルド
- APIソース抽出
- Eclipse インストール＆設定

Ubuntu 10.04 インストール

必要な物

・Ubuntu 10.04 インストールCD
慣れない方には、インストールCD-ROM付きMook本がお勧め

・早くてメモリとHDD容量が大きめのPC
Windowsインストール済でも残りHDD容量があれば可
Mem 2GB, HDD 50GB, CPU マルチコア以上だと幸せになれるかな...



インストール

・インストールは、超簡単
今時のインストーラは、CD-ROMを入れて起動するだけでOK
Windows インストール済みでも、残りHDD容量のパーティション分けまで自動処理

マイSDKを作ってみる

環境設定 （ソース取得＆ビルド）

必要となるツールのインストール

```
$ sudo apt-get install git-core gnupg flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev zlib1g-dev
$ sudo apt-get install valgrind
$ sudo apt-get install libreadline5-dev
```

Java5のインストール

```
$ sudo gedit /etc/apt/sources.list
```

(※)以下を追記して保存終了

```
deb http://us.archive.ubuntu.com/ubuntu/ jaunty multiverse
deb http://us.archive.ubuntu.com/ubuntu/ jaunty-updates multiverse
```

```
$ sudo apt-get update
$ sudo apt-get install sun-java5-jdk

$ sudo gedit /etc/apt/sources.list
```

(※)追記した内容を戻して保存終了

(※)別のバージョンのJavaを入れてる場合、以下コマンドで変更します。
Java6 をインストールしている場合は、 Java5 か Java6 のどちらを選択するかのメニューが出ます。

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

```
$ sudo update-alternatives --config java
  Selection    Path                                        優        Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-6-sun/jre/bin/java        63        auto mode
* 1            /usr/lib/jvm/java-1.5.0-sun/jre/bin/java    53        manual mode
  2            /usr/lib/jvm/java-6-sun/jre/bin/java        63        manual mode

Press enter to keep the current choice[*], or type selection number:
```

参考URL

http://source.android.com/source/download.html Get Android Source Code
http://www.adakoda.com/android/000118.html Androidソースコードをダウンロード・ビルドするには
http://blog.sola-dolphin-1.net/archives/2613458.html Ubuntu-10.04 での Android ビルド環境
http://d.hatena.ne.jp/OkadaHiroshi/20100505/1273073212 Ubuntu 10.04にAndroidの開発環境をインストールしてみた

環境設定 （ソース取得＆ビルド）

git/repo の設定

```
$ mkdir ~/bin
$ curl http://android.git.kernel.org/repo >~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=$PATH:~/bin
$ git config --global user.email ”メールアドレス”
$ git config --global user.name ”ユーザ名”
```

repo と git の概要

・repo とは、androidの長大なソースを効率よくバージョン管理するための、gitラッパーユーティリティのようなもの。
repo は、分散レポジトリ型ソース管理システムである git を内部で使用しています。

・git では、各人がローカルリポジトリ(c)を持ちながら、ワーキングコピー(a)を利用しています。
またソース変更を高速に比較・確認できるよう、索引(b)が使われています。
(※)Git and Repo cheatsheet での呼称：(a).working directory, (b).index, (c).repository

・他人のレポジトリと同期するには、他人のレポジトリと通信を行う必要があります。
・全員が共有するマスタレポジトリを設ければ、単一レポジトリ型と同じような運用が出来ます。

参考URL

http://source.android.com/source/git-repo.html Using Repo and Git
http://groups.google.co.jp/group/android-group-japan/browse_thread/thread/ff3cd64bc222590b
http://b4.x0.com/hiki/?Git%2F%CA%AC%BB%B6%A5%EC%A5%DD%A5%B8%A5%C8%A5%EA%A4%C3%A4%C6%B2%BF%A4%AC%B4%F2%A4%B7%A4%A4%A4%CE
http://www8.atwiki.jp/git_jp/ Git 入門
http://sourceforge.jp/magazine/09/02/02/0655246 分散バージョン管理システムGitの使い方入門
http://sourceforge.jp/magazine/09/03/16/0831212 Gitを使いこなすための20のコマンド

環境設定 （ソース取得＆ビルド）

ビルド環境変数の設定

```
$ export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun  export JAVA_HOME=/usr/lib/jvm/java-6-sun
$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:~/bin
$ export PATH=$JAVA_HOME/bin:$PATH
$ export USE_CCACHE=1
```

```
$ update-java-alternatives -l
java-1.5.0-sun 53 /usr/lib/jvm/java-1.5.0-sun
java-6-sun 63 /usr/lib/jvm/java-6-sun
```

各環境変数の概要

・JAVA_HOME のパスには、Java5 の場合 /usr/lib/jvm/java-1.5.0-sun を設定すると良いでしょう。
update-java-alternatives -l で、インストール済みの Java パッケージの情報が確認できます。
(※)ちなみに私の環境では、/usr/bin/javac は、/etc/alternatives/javac にリンクされ、
さらに、/usr/lib/jvm/java-1.5.0-sun/bin/javac にリンクされていました。

・PATH においては、JAVA_HOME を他のパス設定より優先させるようにします。
/usr/bin や /sbin /usr/sbin が先にあると、そちらに Java があれば優先されるため。
(※)PATH 設定末尾の ~/bin は、repo コマンド用の設定です。

・USE_CCACHE=1 を設定すると、リビルド時にキャッシュが利用されます。

参考URL

https://forums.ubuntulinux.jp/viewtopic.php?id=5970	JAVA_HOME を定義したいがJDK1.6 のインストール先不明
http://d.hatena.ne.jp/embedded/20081106/p1	SDKのビルド
http://d.hatena.ne.jp/hkinami/20081114/p3	make sdk
http://source.android.com/porting/build_system.html	Android Build System

ソースコード取得

ソースコード取得実行コマンド

(※)下記の設定は、froyo ソースコード取得をホーム直下の myfroyo ディレクトリに想定した例です。

```
$ mkdir ~/myfroyo
$ cd ~/myfroyo
$ repo init -u git://android.git.kernel.org/platform/manifest.git -b froyo 2>&1 | tee ../repo_init.log
$ repo sync 2>&1 | tee ../repo_sync.log
```

各実行コマンドの概要

・repo init コマンドは、repo クライアントを初期化(新規生成)します。
実行ディレクトリには、RepoのソースコードとAndroidのマニフェストファイルのためのGitリポジトリが含まれた隠しディレクトリ .repo/ が作成されます。
・-u <url> は、取得するマニフェスト・リポジトリのURL指定です。
・-b <revision> は、対象とするマニフェスト・ブランチのリビジョンを指定します。
-b froyo なら、取得対象がfroyoソースになり、-b オプションがない場合は、master が対象となります。
・Your Name, Your Emailの入力を要求してきますが、取得だけの目的であればブランクでも構わないそうです。

・repo sync コマンドは、リモート・リポジトリとプロジェクト最新ソースの同期を図ります。
sync の後ろに対象とするプロジェクト(のリスト)がない場合は、すべてのプロジェクトが対象となります。
ソース未取得の場合の実行は、リモート・リポジトリからの最新ソース・コピーと同じになります。

(※)repo init と repo sync を連続実行させたい場合は、セミコロン';'で両コマンドを連結(一文化)してください。
(※)2>&1 | tee <ログファイル名> の記述は、標準出力とエラー出力を画面とログに出力させるためだけのものです。

参考URL

http://source.android.com/source/git-repo.html Using Repo and Git
http://www.adakoda.com/android/000118.html Androidソースコードをダウンロード・ビルドするには

マイSDKを作ってみる

SDKビルド

ＳＤＫビルド実行コマンド

(※)下記の設定は、froyo ソースコード取得をホーム直下の myfroyo ディレクトリに想定した例です。

```
$ cd ~/myfroyo
$ source build/envsetup.sh
$ make clean 2>&1 | tee ../make_clean.log
$ make sdk -j4 2>&1 | tee ../make_sdk.log
```

各実行コマンドの概要

・souces build/envsetup.sh は、source コマンドと envsetup.sh シェルスクリプトから成っています。
source コマンドは、シェルスクリプトを source 実行元(呼出元)シェルで実行します。
envsetup.sh シェルスクリプトは、Android ソース・ビルドのための関数や環境変数を設定します。
通常のシェルスクリプト実行は、新規シェルを起動して行われるため、スクリプト中で環境設定をしても、
スクリプト終了と共に新規シェルも終了するため無意味になってしまいますが、source で実行すれば、
自分を呼び出したシェルに環境設定がなされることになります。
(※)source コマンドの代わりに . コマンドを利用して . build/envsetup.sh としても結果は同じです。

・make clean コマンドは、既存のビルド生成物を以下の４ディレクトリから削除します。
out/target/product/generic..., out/target/common..., out/host/linux-x86..., out/host/common...
(※)以前にビルド失敗をしていた場合など、ビルドを最初からやり直したい時に利用します。

・make sdk コマンドは、Android ソースから SDK をビルドします。
-j<num> オプションは、ビルドを複数プロセス？で実行させてコンパイル速度をあげるオプションです。
CPU がシングルコアなら -j2、マルチコア(×2)なら -j4 と自分の環境にあった基数を設定してください。
(※)このオプションを指定しなくても、SDK　のビルドは可能です。

参考URL

http://itpro.nikkeibp.co.jp/article/COLUMN/20060711/242981/　(その１：シェル・スクリプトとは)
http://source.android.com/porting/build_system.html　Android Build System
{Android source root}/devlopement/docs/howto_build_SDK.txt
http://jindroid.com/2009/06/08/howto-build-sdk-from-android-source-code/

マイSDKを作ってみる

SDKビルド

envsetup.sh シェルスクリプトについて

・envsetup.sh シェルスクリプトは、ビルドを行うための一般的な環境設定を行います。
android open source projectの Android Build System章、Device Codeでは、source build/envsetup.sh には、
Android の一般的なビルドを行うために必要となる変数や関数の定義が含まれている旨が記述されています。
(※)envsetup.sh の後で help を実行すれば、利用できる関数一覧が表示されます。
カレントディレクトリ以下をコンパイルする mm 関数などは、JNI を使ったアプリ等の開発に有用です。

```
$ source build/envsetup.sh
including device/htc/dream/vendorsetup.sh
including device/htc/passion/vendorsetup.sh
including device/htc/sapphire/vendorsetup.sh

$ help
Invoke ". build/envsetup.sh" from your shell to add the following functions to your environment:
- croot:   Changes directory to the top of the tree.
- m:       Makes from the top of the tree.
- mm:      Builds all of the modules in the current directory.
- mmm:     Builds all of the modules in the supplied directories.
- cgrep:   Greps on all local C/C++ files.
- jgrep:   Greps on all local Java files.
- resgrep: Greps on all local res/*.xml files.
- godir:   Go to the directory containing a file.
～以降の関数一覧出力を省略～
```

参考URL

http://source.android.com/porting/build_system.html Android Build System
http://d.hatena.ne.jp/hkinami/20081115/p2 . build/envsetup.sh
http://www.swingingblue.net/mt/archives/002611.html androidのenvsetup.shに隠されてる(?) 便利なコマンド
株式会社イーフロー 組み込みプレス Vol.16（技術評論社刊）特集1「Android™ による組込み開発」
http://www.eflow.jp/common/pdf/090828/eflow-android-special-all.pdf

SDKビルド

choosecombo と lunch 関数について

・choosecombo と lunch は、ビルドに関する各種個別情報を設定する envsetup.sh の関数です。
make に関するインターネット上の情報を見ていると、envsetup.sh の後で choosecombo や lunch を利用している例があるので少し補足します。
これらは、対象とする端末種別や使用目的など、ビルドに関する各種個別情報を設定する関数です。
実行によりビルドの対象(*1)とAndroid(*2)に関する環境変数が設定されることになります。
(※)これらを実行しない場合は、デフォルト設定として、PRODUCT(製品種別)に generic(一般的)が、VARIANT(目的種別)に eng(開発版)が、TYPE(提供種別)に release が選択されているようです。

(*1)TARGET_PRODUCT, TARGET_BUILD_VARIANT, TARGET_SIMULATOR, TARGET_BUILD_TYPE
(*2)ANDROID_BUILD_TOP,ANDROID_EABI_TOOLCHAIN, ANDROID_PRODUCT_OUT, ANDROID_QTOOLS, ANDROID_TOOLCHAIN, BUILD_ENV_SEQUENCE_NUMBER

```
$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. generic-eng
     2. simulator
     3. full_dream-userdebug
     4. full_passion-userdebug
     5. full_sapphire-userdebug

Which would you like? [generic-eng] 1
```

```
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=2.2
TARGET_PRODUCT=generic
TARGET_BUILD_VARIANT=eng
TARGET_SIMULATOR=false
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=MASTER
============================================
```

参考URL

http://jindroid.com/2009/06/08/howto-build-sdk-from-android-source-code/
{Android source root}/build/envsetup.sh

SDKビルド

ＳＤＫビルドに成功していたら

・make sdk を実行して、サンプルのようなメッセージが出力されて終了していれば、SDKビルドに成功しています。
生成された SDK は、以下に出力されていますので、マイSDK用のディレクトリにコピーしておきましょう。
SDK出力先：{Android source root}/out/host/linux-x86/sdk/android-sdk_eng.<ユーザ名>_linux-x86

(※)SDKビルド成功時のメッセージ出力最終行サンプル

```
～ これまでのビルドメッセージ省略 ～
Package symbols: out/target/product/generic/generic-symbols-eng.<ユーザ名>.zip
Package SDK: out/host/linux-x86/sdk/android-sdk_eng.<ユーザ名>_linux-x86.zip
```

・ビルドに成功した SDK 内容をマイSDK用のディレクトリにコピー
(※)下記の設定は、マイSDKコピー先としてホーム直下の myfroyo_SDK ディレクトリを想定した例です。

```
cp -r ~/myfroyo/out/host/linux-x86/sdk/android-sdk_eng.<ユーザ名>_linux-x86 ~/myfroyo_SDK
```

・ちなみにビルドに成功したソースと出力結果をアーカイブ保管する場合
展開：tar xvzf ファイル名.tar.gz
圧縮：tar cvzf 圧縮後のファイル名.tar.gz 圧縮前のファイル名(またはディレクトリ名)
(※)下記の設定は、ホーム直下の myfroyo ディレクトリ以下をホーム直下のアーカイブに保管する例です。

```
cd ~/
tar cvzf myfroyo_20100724.tar.gz ~/myfroyo
```

参考URL

http://memorva.jp/memo/linux/tar_gz.php　tar.gzファイルの圧縮・解凍
http://itpro.nikkeibp.co.jp/article/COLUMN/20060228/231198/　【ファイルを圧縮・展開する】

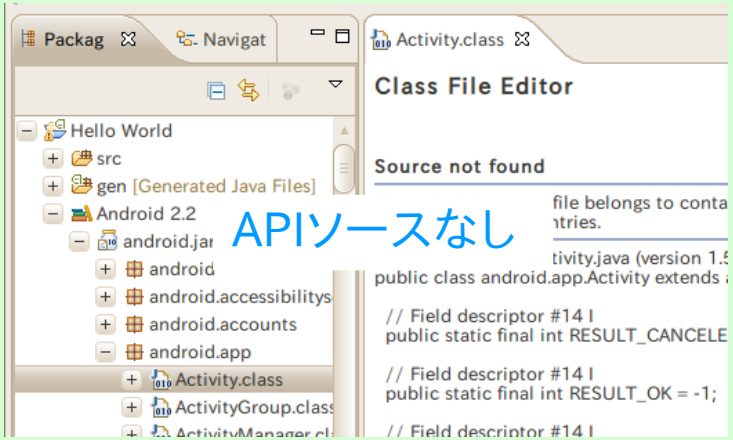
マイSDKを作ってみる

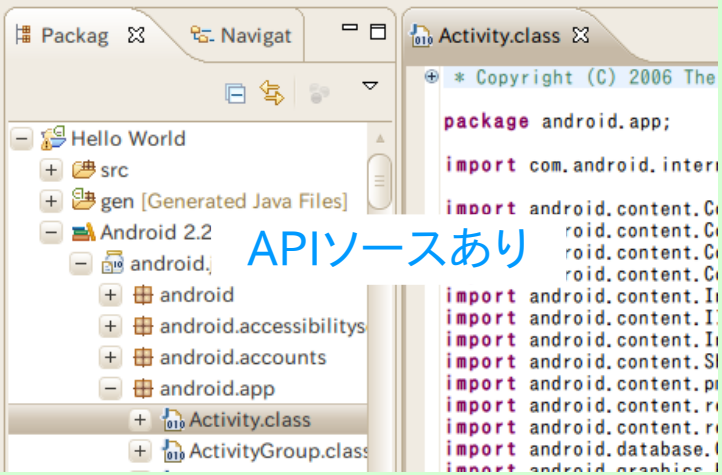
APIソース抽出

APIソースを抽出する理由

・SDK の Android.jar APIライブラリには、ソースが含まれていません。
SDK の Android API ライブラリである Android.jar には、APIソースが含まれていないので、
そのままでは、Eclipse での開発におけるデバッグでのAPI内部のトレースやブレークポイント設定も、
ソース解析でのAPI定義元の F3 参照もできません。

・SDK の Android.jar に API ソースを含ませるには
SDK の platfroms/<Androidバージョン名>/ 配下へ sources という名前でディレクトリを作り、
Android.jar のAPIクラス・パッケージ構造に対応させたソースコード(ソースフォルダ)配置をしてあげる
必要があります。





参考URL

http://blogger.tempus.org/2009/05/eclipse-androidjar.html
http://stuffthathappens.com/blog/2008/11/01/browsing-android-source-in-eclipse/
http://www.grandnature.net/blog/archives/2008/10/android_eclipseandroid_sdk.html
http://magpad.jugem.jp/?eid=114　　EclipseでAndroidのソースコードをたどる方法

APIソース抽出

Androidソースからjavaファイルを抽出する

・Android ソースから java ファイルを抽出するスクリプト・サンプル
幸いマイSDKは、Android ソースから作成したので、Android.jar のクラスの元になったAPIソースがあります。
勉強のためにAPIソース抽出用のスクリプト・サンプルを作ってみました。

(1).ディレクトリ下のすべての java ソースファイルのパスを取得し、
(2).awk スクリプトを起動して
(3)～(7).(1)からテストソースやサンプルを表すディレクトリ(tests,sample)などのパスを除外して、
(8).上記以外のパスに対しては、
(9).java ソース・パスを表す /src/main/java/～*.java のディレクトリとなるルートパスを取得するか、
(10).junit ソース・パスを表す /src/junit/～*.java のディレクトリとなるルートパスを取得するか、
(11)～(17).ソース・ルートがパッケージ先頭を表す 'org' 'com' 'android' 'dalvik' 'javax' 'junit' 'java' の
ディレクトリとなるルートパスを取得し、
(18).(9)～(17)の対象とならなかったパスは、Android システム用のソースではないので除外する。
(19).取得されたソースをソートしてルートパスの重複を除外して、
(20).ホームの'sources'をコピー先に、取得したルートパス下のサブディレクトリを含むファイルコピーを行う。

```
 (1).find . -name *.java |                          (2).awk '
 (3)./^.*tests¥/.*$/{$0="";}                        (4)./^.*¥/sample.*$/{$0="";}
 (5)./^.*¥/test¥/.*$/{$0="";}                       (6)./^.*¥/sdk¥/.*$/{$0="";}
 (7)./^.*¥/out¥/.*$/{$0="";}                        (8)./^{
 (9).sub(/¥/src¥/main¥/java¥/.*¥.java$/,"/src/main/java/*");
(10).sub(/¥/src¥/junit¥/.*¥.java$/,"/src/junit/");
(11).sub(/¥/org¥/.*¥.java$/,"/org/");              (12).sub(/¥/com¥/.*¥.java$/,"/com/");
(13).sub(/¥/android¥/.*¥.java$/,"/android/");      (14).sub(/¥/dalvik¥/.*¥.java$/,"/dalvik/");
(15).sub(/¥/javax¥/.*¥.java$/,"/javax/");          (16).sub(/¥/junit¥/.*¥.java$/,"/junit/");
(17).sub(/¥/java¥/.*¥.java$/,"/java/");            (18).sub(/^.*¥.java$/,"");print;}' |
(19).sort | uniq |
(20).awk '/^¥..*/{src=$0; cmd="cp -R " src " ~/sources"; print cmd; system(cmd);}'
```

APIソース抽出

マイSDKへのAPIソース追加実行コマンド

・マイSDK の Android.jar に、APIのソースを追加する
スクリプト・サンプルは、Android ソースのルート・ディレクトリで(長文ですがワンライナー化して)実行すると、
抽出Javaソースがホーム直下の sources ディレクトリにコピーされるように作りました。
抽出されたAPIソースをマイSDKの platfroms/<Androidバージョン名> 配下へコピーすれば、
マイSDKへの Android.jar ライブラリAPIソースの追加が完了します。

(※)下記は、SDKとAPI抽出ソース配置先をホーム直下の myfroyo_SDK と sources ディレクトリに想定した例です。

```
$ cd ~/myfroyo
$ find . -name *.java | awk '/^.*tests¥/.*$/{$0="";}/^.*¥/sample.*$/{$0="";}/^.*¥/test¥/.*$/
{$0="";}/^.*¥/sdk¥/.*$/{$0="";}/^.*¥/out¥/.*$/{$0="";}//
{sub(/¥/src¥/main¥/java¥/.*¥.java$/,"/src/main/java/*");sub(/¥/src¥/junit¥/.*¥.java$/,"/src/junit/");su
b(/¥/org¥/.*¥.java$/,"/org/");sub(/¥/com¥/.*¥.java$/,"/com/");sub(/¥/android¥/.*¥.java$/,"/android/");s
ub(/¥/dalvik¥/.*¥.java$/,"/dalvik/");sub(/¥/javax¥/.*¥.java$/,"/javax/");sub(/¥/junit¥/.*¥.java$/,"/jun
it/");sub(/¥/java¥/.*¥.java$/,"/java/");sub(/^.*¥.java$/,"");print;}' | sort | uniq | awk '/^¥..*/
{src=$0; cmd="cp -R " src " ~/sources"; print cmd; system(cmd);}'
$ cp -r ~/sources ~/myfroyo_SDK/android-sdk_eng.<ユーザ名>_linux-x86/platforms/android-2.2
```

(※)API抽出実行中の例です。

```
cp -R ./build/tools/apicheck/src/com/ ~/sources
cp -R ./cts/tools/cts-reference-app-lib/src/android/ ~/sources
cp -R ./cts/tools/host/src/com/ ~/sources
cp -R ./dalvik/dx/src/com/ ~/sources
cp -R ./dalvik/dx/src/junit/ ~/sources
cp -R ./dalvik/hit/src/com/ ~/sources
cp -R ./dalvik/libcore-disabled/instrument/src/main/java/* ~/sources
～ 省略 ～
```

Eclipse インストール＆設定

SDK環境設定および Eclipse のダウンロードとインストール

・android developer サイト内参考先
サイトの各章(*1)を参考に、SDK環境設定および、Eclipse のダウンロードやインストールから
ADT(Android Development Tools)の設定を行ないます。

(*1)SDK System Requirements　http://developer.android.com/sdk/requirements.html
Installing the SDK　http://developer.android.com/sdk/installing.html
ADT Plugin for Eclipse　http://developer.android.com/sdk/eclipse-adt.html

・Eclipse のダウンロードとインストール
Eclipse.org homeの Downloadページ(http://www.eclipse.org/downloads/)から、
最新版の Eclipse 3.6(Helios) Linux 32Bit版、Eclipse IDE for Java Developers(98MB)
eclipse-java-helios-linux-gtk.tar.gz を選択かつダウンロードして、
ホーム直下の bin というディレクトリに展開するだけでインストール終了です。
展開後のホーム直下には、~/bin/eclipse というディレクトリができ、その下に本体バイナリが配置されます。

(※)下記は、ホーム直下の Download ディレクトリの eclipse圧縮ファイルをホーム直下の bin に展開する例です。

```
$ cd ~/bin
$ tar zxvf ~/Downloads/eclipse-java-helios-linux-gtk.tar.gz
```

(※)Ubunts 10.04 のパッケージ・マネージャーで管理している eclipse バージョンは、3.5.2 の様です。(2010/07/24調べ)
(※)android developers の SDK System Requirements章、Supported Development Environments Eclipse IDE では、
3.4(Ganymede)か 3.5(Galileo) を対象としています。　注意事項として Eclipse 3.6(Helios)に、
ADT プラグインの不具合があるので 3.5 に留まるよう勧めています。

・Eclipse 起動（環境変数設定）
あらかじめマイSDKと Eclipse のパスを設定しておいてから、eclipse を起動します。

```
$ export ANDROID_SDK_HOME=~/myfroyo_SDK/android-sdk_eng.<ユーザ名>_linux-x86
$ export ECLIPSE_HOME=~/bin/eclipse
$ export PATH=$PATH:$ANDROID_SDK_HOME/tools
$ export PATH=$PATH:$ECLIPSE_HOME
$ eclipse
```

Eclipse インストール＆設定

ＡＤＴプラグイン・インストールとＳＤＫロケーション設定

・ADT プラグインのインストール
Eclipse 3.6(Helios)での ADT プラグイン・インストール手順

1. Eclipse のメニューから Help > Install New Software を選びます。
2. Install ダイアログの Available Software が開くのでリポジトリを追加する[Add...]ボタンをクリックします。
3. 更に Add Repository ダイアログが開くので Name に ”Android Plugin”を
   Location に ”https://dl-ssl.google.com/android/eclipse/”を設定し、[OK]ボタンをクリックして閉じます。
4. Avaliable Software ダイアログに戻りますので、Work with: ドロップダウンリストから
   ”Android Plugin - https://dl-ssl.google.com/android/eclipse/”を選択して、
   インストールするソフトウェア一覧を下段に表示させます。
5. インストールするソフトウェア一覧に”Developer Tools”が表示されるので[Select All]ボタンをクリックします。
   (※)”Developer Tools”と、配下の”Android DDMS”と”Android Development Tools”にチェックが入ります。
6. 最下段の[Next>]ボタンをクリックして、次ページ(Install Details)に移りましたら[Finish]をクリックします。
7. Eclipse を再起動します。

・SDK Location の設定
Eclipse 3.6(Helios)での SDK ロケーション設定手順

1. Eclipse のメニューから Window > Preferences を選んで、Prefernces パネルを開きます。
2. パネル左側の Android を選択して、Android の設定パネルに移ります。
3. SDK Location にマイSDKの配置先
   ”/home/<ユーザ名>/myfroyo_SDK/android-sdk_eng.<ユーザ名>_linux-x86”を設定します。
4. [Apply]ボタンをクリックし、[OK]ボタンをクリックしましたら設定完了です

その他

マイSDK用の環境変数設定

・ソースコード取得からSDKビルドおよび Eclipse設定までの各種環境変数設定を毎回繰り返すのは、面倒です。
以下の「環境変数設定内容サンプル」を ~/.bashrc に追加(*1)するか、mysdk_console.sh とか名した
専用のシェルスクリプト(*2)に追加(新規作成)しておけば便利と思います。

(*1).bashrc に設定した場合、すべてのターミナル(端末)で環境変数が設定済みとなります。
(*2)mysdk_console.sh を作成した場合は、ターミナル(端末)を開いてから
$source mysdk_console.sh を実行してください。

(※)環境変数設定サンプル

```
export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun
#export JAVA_HOME=/usr/lib/jvm/java-6-sun
export USE_CCACHE=1
export ANDROID_SDK_HOME=~/myfroyo_SDK/android-sdk_eng.<ユーザ名>_linux-x86
export ECLIPSE_HOME=~bin/eclipse
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
export PATH=$JAVA_HOME/bin:$PATH
export PATH=$PATH:$ANDROID_SDK_HOME/tools
export PATH=$PATH:$ECLIPSE_HOME
export PATH=$PATH:~/bin
```

マイSDKを作ってみる

お疲れさまでした
これでマイSDKの説明は、
すべて終了です。

と…したかったのですが

本当の苦労は、次ページからなのです…

マイSDKを作ってみる

おわび

ソースにパッチをあてないとEclipse開発時にエラーが...orz

・Eclipse開発時に発生する問題
ごめんなさい、SDK ビルドがエラーなく終了したからといって安心できないことが判りました。
repo ツールで froyo ブランチから(2010/07/24に)取得した最新ソースを SDK ビルドしてみると、
エラーなく終了(作成されたSDKツールでエミュレータ起動もできます)するのですが、
Eclpce に組み込んで Hello World アプリを作成 > コンパイルしてみると、Console パネルに
リソース・クラス作成ができないというエラーメッセージが出力されたのです。

```
public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Problems | @ Javadoc | Declaration | Console

Android

```
[2010-07-31 00:13:02 - Hello World] W/ResourceType( 4659): Unable to get
[2010-07-31 00:13:02 - Hello World] /home/rie/workspace/Hello World/res/
```

```
] W/ResourceType( 4659): Unable to get buffer of resource asset file
] /home/rie/workspace/Hello World/res/layout/main.xml:2: error: No resourc
] /home/rie/workspace/Hello World/res/layout/main.xml:2: error: No resourc
] /home/rie/workspace/Hello World/res/layout/main.xml:2: error: No resourc
] /home/rie/workspace/Hello World/res/layout/main.xml:7: error: No resourc
```

おわび

ソースにパッチをあてないとEclipse開発時にエラーが...orz

・問題を回避するためのソースへのパッチ
調査の結果、バッファサイズが小さいために当該エラーが発生する様です。(詳細は下記)

(※)ADT のプリ・コンパイラは、R.java ファイルを作成するために aapt(Android Asset Packaging Tool)という
リソースをコンパイルして assets に入れてくれるツールを利用するそうですが、
aapt の出力バッファよりも大きいサイズのリソースを扱おうとするとき(*1)に当該エラーが発生するようです。

(*1)ADT 0.9 update で同じ問題が発生したときは、Android.jar に含まれる resources.arsc と呼ばれる
圧縮リソースファイルのサイズが大きかったそうです。

この問題を急いで回避するためには、バッファ取扱いサイズを変更するパッチを
{Android root}/frameworks/base/include/utils/asset.h に当てると良いそうです。

(※)オリジナル

```
#ifdef HAVE_ANDROID_OS
        UNCOMPRESS_DATA_MAX = 1 * 1024 * 1024
#else
        UNCOMPRESS_DATA_MAX = 2 * 1024 * 1024
#endif
```

→

(※)変更後例

```
#ifdef HAVE_ANDROID_OS
        UNCOMPRESS_DATA_MAX = 2 * 1024 * 1024
#else
        UNCOMPRESS_DATA_MAX = 4 * 1024 * 1024
#endif
```

参考URL

After the ADT 0.9 update, Eclipse can not create R.java automatically
http://groups.google.com/group/android-developers/browse_thread/thread/6192b2822bc369df/a789094adf1902b9?#a789094adf1902b9
http://developer.android.com/guide/developing/tools/aapt.html Using aapt
http://d.hatena.ne.jp/itog/20091120 Android SDKの生成、自作SDKを使ったアプリ開発

マイSDKを作ってみる

おわび

SDKビルドでは、カメラ廻りの設定が足りないようです

・ビルドしたSDKでは、カメラ廻りの設定が足りないようです
サンプルの ApiDemos を(Eclipse プロジェクトとしてインポートして)実行すると、android.hardware.camera が必要なのに利用できない(*1)(*2)とされて、インストールに失敗しアプリ起動できません。

(※)エラーについて調べてみると、ApiDemos の実行には、Android ソースに USE_CAMERA_STUB := true という設定を必要としますが、前記設定がないためエラーが発生していると思われます。
(※)別件ですが Camera アプリを実行させると Sorry! ダイアログがポップアップして強制終了を求められます。

```
(*1)Console 出力
[2010-07-24 11:43:56 - ApiDemos] Installing ApiDemos.apk...
[2010-07-24 11:44:05 - ApiDemos] Installation error: INSTALL_FAILED_MISSING_FEATURE
[2010-07-24 11:44:05 - ApiDemos] Please check logcat output for more details.
[2010-07-24 11:44:05 - ApiDemos] Launch canceled!
```

```
(*2)LogCat 出力
07-24 02:44:03.025: ERROR/PackageManager(59): Package com.example.android.apis requires unavailable
feature android.hardware.camera; failing!
```

参考URL

No actions in intent filter at /data/app/ApiDemos.apk
http://groups.google.co.jp/group/android-porting/browse_thread/thread/55b1a1236148d1db

マイSDKを作ってみる

おわび

SDKビルドでは、カメラ廻りの設定が足りないようです

・ApiDemos をインストールさせて起動できるようにする設定

(1).ApiDemos の AndroidManifest.xml を編集する方法　(ApiDemosのandroid.hardware.camera要求を外す)

ApiDemos をインストールさせ起動できるようにするには、ApiDemos の AndroidManifest.xml にある、”android.hardware.camera”についてのタグ部(青色表記)を以下の記述のようにコメントアウトします。

```
(※)AndroidManifest.xml 変更例
<uses-permission android:name="android.permission.CAMERA" />
<!-- <uses-feature android:name="android.hardware.camera" /> -->
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
```

(2).SDKビルド対象を任意の端末に指定する方法 (端末指定で USE_CAMERA_STUB =: true の設定を行わせる)

SDKビルド時に lunch メニューを起動して、ターゲット端末を generic(一般的)でなく任意端末に指定します。

(※)任意端末を指定するSDKビルド例

```
$ cd ~/myfroyo
$ source build/envsetup.sh
$ lunch 3
$ make sdk -j4 2>&1 | tee ../make_sdk.log
```

lunch 3 を実行することで、TARGET_PRODUCT を full_dream、TARGET_BUILD_VARIANTを userdebug に設定します

SDKビルドの lunch 関数の実行例を参照されれば、”3. full_dream-userdebug”とありますように、

lunch 3 により 端末を htc-dream(ADP1)に指定します。

(※)(1)(2)いずれの場合でも、Camera アプリ実行時の Sorry! ダイアログは抑止できません。

参考URL

No actions in intent filter at /data/app/ApiDemos.apk

http://groups.google.co.jp/group/android-porting/browse_thread/thread/55b1a1236148d1db

おわび

Mac OS および 64-bit x86 の皆様へ

私(robo)の開発環境が 32-bit x86 のため動作確認ができず、
放置のままで申し訳ありません。

android open source project サイトの
Get Android Source Code章、Setting up your machine を参考に、
開発環境設定の「必要となるツールの取得」を変更すれば、
確証はありませんが応用可能かと思います。

この発表情報は、既に古くなっているかもしれません

Android ソースは、オープンソースのため日々変わっていきます。
この発表情報は、2010/07/24に froyo ブランチから取得したソースについての
SDKビルドの旨、御了承ください。

参考URL
http://source.android.com/source/download.html Get Android Source Code

ご 静 聴

ありがとうございました