Yet another * programming blog

for * in Linux, Android, Kernel, Open Source

How to build a custom Android emulator image

If you want to play with Android and do not have any hardware device, your best option is to use the emulator. Depending on the degree of changes you want to make in the source code, you have several options. You can just download the emulator and run it or compile it from the Google source code. Since the source code comes with a pre-compiled kernel you will need to get the kernel separately and build it if you want to make changes there.

Run the Android emulator

The Android Emulator can be downloaded with the Android SDK. Google has detailed and up to date information on their developer site on how to this.

You first need to create an AVD configuration. Then you can run the emulator using:

```
$ emulator -avd my_avd
```

Build the Android emulator from source

First thing you need to do is to download the Android source tree from Google. If you do not need a specific version, it is best to download the master branch to get all the latest changes and fixes. In a nutshell, this is what you have to do:

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ mkdir -p ~/workspace/android/google
$ cd ~/workspace/android/google
$ repo init -u https://android.googlesource.com/platform/manifest
$ repo sync
```

Next you need to build the emulator. You can build the emulator for arm or x86. Since the arm emulator will run quite slow on you machine, it is best to build the x86 emulator.

Build the x86 emulator

```
$ source build/envsetup.sh
$ lunch full_x86-eng
$ make -j4
```

Now you can go get some tea or coffee cause this will take a while depending on the machine you're building on. You can improve the build time by using ccache.

When the building finishes, run the emulator:

```
$ emulator -wipe-data &
```

You can specify various parameters to the emulator. By -wipe-data you reset the user data image. It is not necessary to use this parameter every time, but I find it a good practice to do so. I sometimes have problems with the emulator not starting and using -wipe-data fixes this.

Build the arm emulator

If for some reason you need to build the emulator for arm here are the steps:

```
$ source build/envsetup.sh
$ lunch full-eng
$ make -j4
$ emulator -wipe-data &
```

Build the emulator kernel from source

Now you have the emulator running and you also have the source tree. You can tweak any part of the code, recompile and test your changes in the emulator. What happens if you need to change the kernel config? Maybe you want to try out some kernel debugging options and these are not enabled in the prebuilt kernel image (true story – it happened to me).
You need to download and build the kernel for Android separately.

The kernel for the emulator is the goldfish kernel. You can download it using:

```
$ cd ~/workspace/android
$ git clone https://android.googlesource.com/kernel/goldfish.git
$ cd goldfish
```

You may notice that you do not have any files in this directory after git clone. This happens because the master branch does not contain any code. You need to checkout a specific branch for the kernel. To check what branches are available you use git branch -a:

```
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/android-goldfish-2.6.29
remotes/origin/android-goldfish-3.4
remotes/origin/linux-goldfish-3.0-wip
remotes/origin/master
```

The stable and working branch is 2.6.29. You need to checkout this branch:

```
$ git checkout -b 2.6.29 origin/android-goldfish-2.6.29
```

Now you have the source code and you can start building the kernel.

Build the kernel for the x86 emulator

Before you start building your kernel, you need to set the cross-compiler path. Since you already downloaded the source tree you have the toolchain there.

```
$ export CROSS_COMPILE=${ANDROID_BUILD_TOP}/external/qemu/distrib/kernel-toolchain/android-kernel-toolchain-
$ export REAL_CROSS_COMPILE=${ANDROID_BUILD_TOP}/prebuilts/gcc/linux-x86/x86/i686-linux-android-4.6/bin/i686-linux-android-
```

${ANDROID_BUILD_TOP} should point to the top of your Google Android tree. It should be set automatically if you have the Google environment set. In order to do that you need to run source build/envsetup.sh and lunch full_x86-eng.

Next you need to set the architecture, configure and build the kernel:

```
$ export ARCH=x86
$ export SUBARCH=x86
$ make goldfish_defconfig
$ make
```

In order to start the emulator using the newly compiled kernel you must use the -kernel option:

```
$ cd ${ANDROID_BUILD_TOP}
$ emulator -kernel ~/workspace/android/goldfish/arch/x86/boot/bzImage -wipe-data &
```

Build the kernel for arm emulator

If you need to build for arm, this is what you have to do:

```
$ export CROSS_COMPILE=${ANDROID_BUILD_TOP}/prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-4.6/bin/arm-linux-androideabi-
$ export ARCH=arm
$ export SUBARCH=arm
$ make goldfish_armv7_defconfig
$ make
$ cd ${ANDROID_BUILD_TOP}
$ emulator -kernel ~/workspace/android/goldfish/arch/arm/boot/zImage -wipe-data &
```

You may try to use make goldfish_defconfig insteaf of goldfish_armv7_defconfig. Currently the kernel built with goldfish_defconfig does not boot.

This entry was posted in Android and tagged arm, emulator, emulator-arm, emulator-x86, google, kernel, qemu, x86 on September 22, 2012 [https://yaapb.wordpress.com/2012/09/22/build-a-custom-android-emulator-image/] .