ESC SV 2012 Embedded Android Workshop part 2

Run your own build

Assuming your operating directory is “~/Desktop/android”

Preserve old copy of default Bone images

```
$ cp -a BeagleBone BeagleBone-orig
```

Create new rootfs images

```
$ cd TI_Android_GingerBread_2_3_4_AM335x_Sources/out/target/product/beagleboard
$ mkdir android_rootfs
$ cp -a root/* android_rootfs/
$ cp -r system android_rootfs/
$ sudo ../../../../build/tools/mktarball.sh ../../../host/linux-x86/bin/fs_get_stats android_rootfs . rootfs rootfs_beaglebone.tar.bz2
```

Copy new rootfs to BeagleBone directory

```
$ sudo cp rootfs_beaglebone.tar.bz2 ~/Desktop/android/BeagleBone/Filesystem
```

Plug the MicroSD card into the MicroSD-USB connector and into your laptop

Share the USB device with the guest VM

VirtualBox->Devices->USB Devices->MicroSD card

Program the MicroSD card

```
$ sudo ./mkmmc-android.sh /dev/sdb
```

Put the SD card back into the Bone, reboot it and connect through VNC as we did earlier

Check the version of Android you're running, it should be time-stamped with the date of your build

apps->Settings->About Phone->Android version

Put HelloWorld in AOSP

For some of the rest of the exercises you'll need to slides from earlier this week

http://www.opersys.com/blog/esc-sv-2012-ea

Copy your HelloWorld from the Eclipse workspace directory to [aosp]/packages/apps

Add an Android.mk that contains the following:

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(call all-java-files-under, src)
LOCAL_PACKAGE_NAME := HelloWorld

include $(BUILD_PACKAGE)
```

Add HelloWorld to [aosp]/build/product/core.mk

Rebuild the AOSP

DON'T “make clean”

Just what we did earlier

```
$ make TARGET_PRODUCT=beaglebone OMAPES=4.x -j4
```

Regenerate the rootfs

Reflash the SD card

Restart the Bone

Connect through VNC

Make sure your HelloWorld is in the app launcher

Start your HelloWorld by clicking on it

Shell into the Bone using ADB

Use the “am” command on the command line to start your HelloWorld

```
# am start -n com.foo.bar/.HelloWorldActivity
```

Disable phone signal icon from the status bar

Open the following file

[aosp]/frameworks/base/packages/SystemUI/src/com/android/systemui/statusbar/StatusBarPolicy.java

Look for 4 instances of

```
mService.setIcon("phone_signal", ...
```

Add the following line after each instance

```
mService.setIconVisibility(“phone_signal", false);
```

Rebuild the AOSP

Regenerate the rootfs

Reflash the SD card

Restart the Bone

Connect through VNC

Make sure your status bar no longer has a phone signal

Temporarily disable the Zygote

Shut the Bone down.

From minicom type:

```
# reboot
```

As the board is rebooting, interrupt U-Boot by typing ENTER

Remove the MicroSD card from the Bone

Reconnect the MicroSD card to Ubuntu

The card's partitions should remount automatically on Ubuntu

Open the following file from the “rootfs” partition

/init.rc

Disable the start of the Zygote (see addition in bold)

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
    socket zygote stream 666
    onrestart write /sys/android_power/request_state wake
    onrestart write /sys/power/state on
    onrestart restart media
    disabled
```

Save file modification

Unmount sd card

Restart the Bone
Connect through VNC
You should NOT see anything come up BUT a green robot
Shell into the Bone using ADB
Start the zygote

```
# start zygote
```

Watch the zygote come up

```
# logcat
```

You should remove the “disabled” keyword from the init.rc for future step

Glibc-based rootfs

Extract the gibc rootfs
Modify AOSP build system to copy content of your rootfs to its default ramdisk. You will need to:

Amend [aosp]/system/core/rootdir/Android.mk to make it look like this (a. additions are in bold, b. don't forget to use TABS, not spaces for the “my_dir” make target's command, c. PATH_TO_MY_GLIBC_ROOTFS is a placeholder you need to replace with the location of the filesystem created in the exercises of the previous section):

```
...
    $(TARGET_OUT_DATA)

my_dir:
    cp -a $(PATH_TO_MY_GLIBC_ROOTFS)/* $(TARGET_ROOT_OUT)

$(DIRS): my_dir
...
```

Amend [aosp]/system/core/include/private/android_filesystem_config.h's “static struct fs_path_config android_files[]” to add an entry for “lib/*” so that the execute bit is preserved for all files in that directory.

Modify the AOSP so that ADB uses BusyBox' shell instead of the default Android shell

ADB is in [aosp]/system/core/adb
You need to modify the SHELL_COMMAND macro in services.c

```
#if ADB_HOST
#define SHELL_COMMAND "/bin/sh"
#else
//#define SHELL_COMMAND "/system/bin/sh"
#define SHELL_COMMAND "/bin/sh"
#endif
```

Also make sure the system path ($PATH) points to the BusyBox “binaries”. The path setting line in your init.rc should then look something like:

```
export PATH /bin:/sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin :/usr/sbin
```

Rebuild the AOSP
Regenerate the rootfs
Reflash the SD card
Restart the Bone
Shell into the Bone using ADB
You should now have a color-coded “ls” and your shell should start with “/ #” instead of just “#”