Android Device Porting Walkthrough

Benjamin Zores

ABS 2012 – 13th February 2012 – Redwood Shores, SF Bay, CA, USA

THE LINUX FOUNDATION

Alcatel·Lucent

Android Device Porting Walkthrough
About Me

ALCATEL LUCENT

SOFTWARE ARCHITECT

- Expert and Evangelist on Open Source Software.
- 9y experience on various multimedia/network embedded devices design.
- From low-level BSP integration to global applicative software architecture.

OPEN SOURCE

PROJECT FOUNDER, LEADER AND/OR CONTRIBUTOR FOR:

• OpenBricks	Embedded Linux cross-build framework.
• GeeXboX	Embedded multimedia HTPC distribution.
• Enna	EFL Media Center.
• uShare	UPnP A/V and DLNA Media Server.
• MPlayer	Linux media player application.

EMBEDDED LINUX CONFERENCE

FORMER ELC EDITIONS SPEAKER

• ELC 2010	GeeXboX Enna: Embedded Media Center.
• ELC-E 2010	State of Multimedia in 2010 Embedded Linux Devices.
• ELC-E 2011	Linux Optimization Techniques: How Not to Be Slow ?

Android Device Porting Walkthrough
My Experience

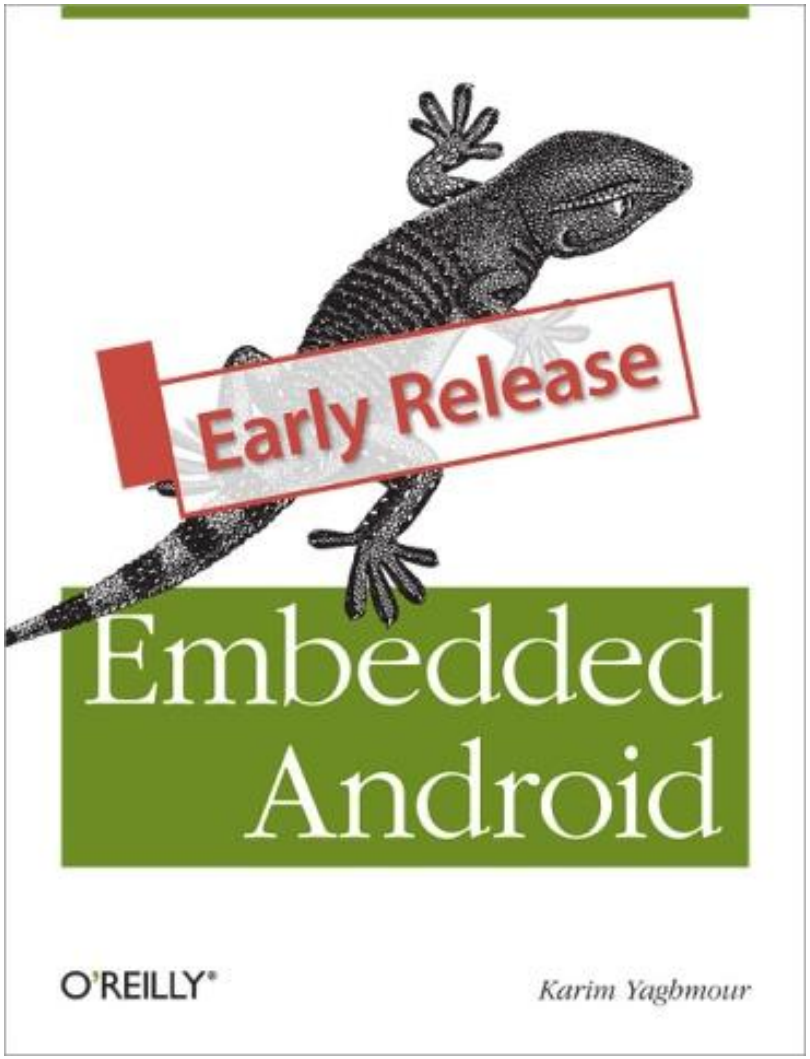
- Expert in Linux Embedded Systems and Cross-Build Systems.

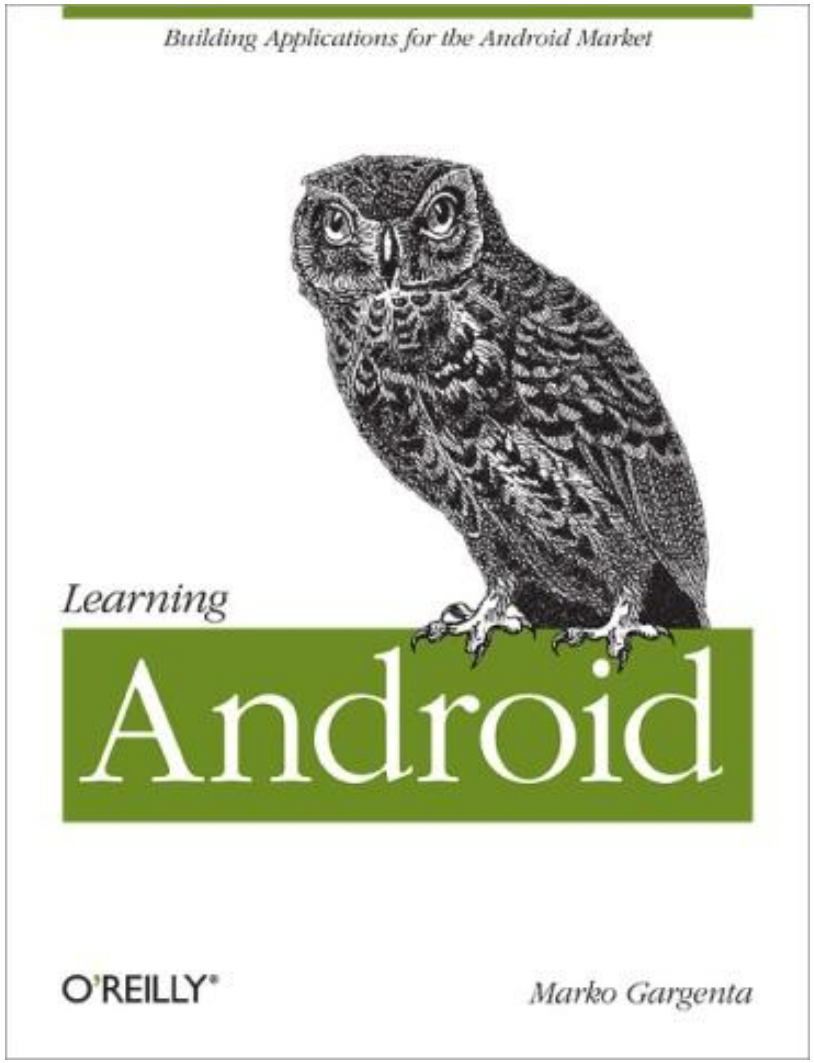
- Used to work on various embedded arch and Vendors:
  - x86 (AMD Geode, Intel Tolapai and CE4xxx).
  - PowerPC (Freescale 6xx, e300 and e500 cores).
  - MIPS 64 (Cavium Octeon CN52xx).
  - ARM:
    - XScale (Intel PXA 250).
    - ARM 9 (Marvell Orion and Kirkwood).
    - ARM 11 (Broadcom Bcmring).
    - Cortex-A8 (TI OMAP 3 and Freescale i.MX5x).
    - Cortex-A9 (TI OMAP 4, Freescale i.MX6 and nVidia Tegra2).

Newbie in Android though learning more about it every day ...

Android Device Porting Walkthrough

My Android Bibles





Android Device Porting Walkthrough
Work Context

- Designing an **Enterprise Desktop IP Phone**.

- Differs heavily from usual Android devices:
  - Always connected, no battery
  - No GSM/CDMA Phone/Modem
  - Always docked
  - No screen rotation
  - No accelerometer
  - No GPS
  - **=> Not a Smartphone**

Android Device Porting Walkthrough
Ice Cream Sandwich Device Compatibility Guidelines

- **Goals**
  - Ensure Google device certification.
  - Provide Android MarketPlace support.
  - Provide Android applications compatibility.
  - Provide access to Google services (GMail, Calendar ...).

- **Key Requirements**
  - Memory: Minimum of 512 MB.
  - Storage: Minimum of 2 GB.
  - Display: Minimum resolution of 460x320.
  - Graphics: Hardware 2D/3D engine is more than recommended.

  - DO NOT modify Google APIs and framework.
  - Android "Telephony" refers to 3G/CDMA, hence IP-Phones are **NOT** considered as Android phones.

Android Device Porting Walkthrough
Which Android Sources to Start With ?

- **Google**
  - Up-to-date reference but limited devices support (reference design only).
  - https://android.googlesource.com/platform/manifest
- **Linaro**
  - Most integrated with wide hardware support.
  - git://android.git.linaro.org/platform/manifest.git
- **Cyanogen Mod**
  - Most features but mostly used for tuning already released commercial products.
  - https://github.com/cyanogenmod
- **Vendor BSP**
  - Potentially outdated but best support for a given platform/SoC.

We'll base our example on TI OMAP4 Pandaboard-derived board based on Linaro sources.

Android Device Porting Walkthrough
Linaro Android Added-Value

- **Extra Packages:**
  - x264, ffmpeg, powertop, ARM DS-5 Gator Profiler, U-Boot, Linux Kernel, BusyBox
- **Optimized Packages:**
  - Dalvik, IPSec, libvpx, libpng, libjpeg-turbo, OpenSSL, GCC Toolchain
- **Proprietary-Open Firmwares (BT, WiFi ...)**
- **More hardware devices support than stock Google ICS:**
  - TI Pandaboard
  - TI BeagleBoard
  - Freescale i.MX53
  - ST Snowball
  - Samsung Origen

Android Device Porting Walkthrough
AOSP Custom Device HowTo

- Create your own **device/my_company/my_device** directory with complete product description.
- Mandatory **vendorsetup.sh**:

```
add_lunch_combo my_device-eng
```

- Mandatory **Android.mk**:

```
LOCAL_PATH := $(call my-dir)
include $(call all-makefiles-under,$(LOCAL_PATH))
```

- Mandatory **AndroidProducts.mk**:

```
PRODUCT_MAKEFILES := $(LOCAL_DIR)/my_device_name.mk
```

- Mandatory **my_device_name.mk**:

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/full_base.mk)
$(call inherit-product, device/my_company/common/common.mk)
$(call inherit-product, device/my_company/my_device/device.mk)

PRODUCT_BRAND                   := my_device_brand
PRODUCT_DEVICE                  := my_device_name
PRODUCT_NAME                    := my_device_name
```

Android Device Porting Walkthrough
AOSP Custom Device HowTo

- Mandatory **device.mk**:

```
PRODUCT_COPY_FILES     += device/my_company/my_device/init.rc:root/init.rc
PRODUCT_PACKAGES               += audio.primary.omap4
PRODUCT_PROPERTY_OVERRIDES  += hwui.render_dirty_regions=false
PRODUCT_CHARACTERISTICS         := tablet,nosdcard
PRODUCT_TAGS                        += dalvik.gc.type-precise

$(call inherit-product, frameworks/base/build/tablet-dalvik-heap.mk)
$(call inherit-product-if-exists, device/ti/proprietary-open/install-binaries.mk)
```

- Mandatory **BoardConfig.mk** (product-specific compile-time definitions):
  - See **build/core/product.mk** for a list of nice-to-know build options.
  - **# Platform**
    TARGET_BOARD_PLATFORM := omap4

Android Device Porting Walkthrough
AOSP Custom Device HowTo

- **# Target and compiler options**

```
TARGET_CPU_ABI                          := armeabi-v7a
TARGET_CPU_ABI2                         := armeabi
TARGET_CPU_SMP                          := true
```

- **# Enable NEON feature**

```
TARGET_ARCH_VARIANT                     := armv7-a-neon
ARCH_ARM_HAVE_TLS_REGISTER              := true
TARGET_EXTRA_CFLAGS       += $(call cc-option,"-march=armv7-a -mtune=cortex-a9",
                             $(call cc-option,"-march=armv7-a -mtune=cortex-a8"))
```

- **# Filesystem and partitioning**

```
BOARD_SYSTEMIMAGE_PARTITION_SIZE        := 256M
BOARD_USERDATAIMAGE_PARTITION_SIZE      := 512M
BOARD_CACHEIMAGE_PARTITION_SIZE         := 256M
BOARD_CACHEIMAGE_FILE_SYSTEM_TYPE       := ext4
BOARD_FLASH_BLOCK_SIZE                  := 4096
TARGET_USERIMAGES_USE_EXT4              := true
TARGET_USERIMAGES_SPARSE_EXT_DISABLED   := true
```

- **# System**

```
TARGET_NO_RECOVERY                      := true
TARGET_PROVIDES_INIT_RC                 := true
```

Android Device Porting Walkthrough
AOSP Custom Device HowTo

- Configuration Overlay
  - In **device.mk**:

```
DEVICE_PACKAGE_OVERLAYS      := device/my_company/my_device/overlay
```

  - Create a **device/my_company/my_device/overlay** directory with same depth and files you want to overwrite. e.g:
    - frameworks/base/core/res/res/values/config.xml
    - frameworks/base/core/res/res/values/dimens.xml
    - frameworks/base/core/res/res/xml/storage_list.xml

Android Device Porting Walkthrough
Bootloader

- Google provides one in **bootable/bootloader/legacy**.
- But usually replaced by U-Boot with Fastboot protocol support.
- Can be either out of AOSP sources or integrated.
- In **BoardConfig.mk**:
  - TARGET_USE_XLOADER := false
    XLOADER_CONFIG := omap4430panda_config
    TARGET_NO_BOOTLOADER := true # Uses u-boot instead
    TARGET_USE_UBOOT := true
    UBOOT_CONFIG := omap4_panda_config

Android Device Porting Walkthrough
Linux Kernel Androidisms

- **Kernel Androidisms:**
  - **Ashmem**, shared memory allocator.
  - **Binder** IPC and RMI system.
  - **Pmem**, process memory allocator.
  - **Logger**, system logging facility.
  - **Wakelocks**, power management and suspend.
  - **Low Memory Killer**, OOM Tuning.
  - **Alarm Timers**.
  - **Paranoid Network Security**.
  - **Timed GPIO**.
  - **RAM Console**.

- See http://elinux.org/Android_Kernel_Features for more details.

Android Device Porting Walkthrough
Linux Kernel Drivers Policy

- Usually no drivers built as modules:
  - No udev.
  - Smaller disk footprint.
  - Faster to load up.
  - Kernel is built for a static hardware configuration.
- Except for proprietary drivers (usually) or those requiring firmware (e.g. WiFi).
- Can be added to Android FS separately as for firmwares.
- In **BoardConfig.mk**:

```
BOARD_KERNEL_BASE          := 0x80000000
BOARD_KERNEL_CMDLINE       :=
TARGET_NO_KERNEL           := false
KERNEL_CONFIG              := android_omap4_defconfig
```

Hint: Develop and debug raw drivers on Linux OS, not Android.

Android Device Porting Walkthrough
Bionic C Library

- Small footprint BSD-licensed C library with the following weirdness:
  - Non POSIX.
  - No support for locales.
  - No support for wide chars (i.e. multi-byte characters).
  - Comes with its own smallish implementation of pthreads based on Linux futexes
  - C++ exceptions are not supported.
  - pthread cancellation is NOT supported.
  - Several functions are just stubs (i.e. runtime weirdness may happen).
  - Non standard /etc config files:
    - Dynamic user/groups management (no pam, group, passwd or shadow)
    - No fstab
    - No SysV init.

Android Device Porting Walkthrough
System Properties and Settings Databases

- In-memory Bionic System Properties management
  - Array of volatile **property=value** fields.
  - Can be get/set through **getprop/setprop** shell commands as well as Java API.
  - No documentation on existing properties.
  - Anyone can add his own custom properties.

- More persistent configuration settings are available in SQLite databases:
  - e.g: **/data/data/com.android.providers.settings/database/settings.db**
  - See **secure** and **system** tables.

Android Device Porting Walkthrough
Android Init

- Proprietary init language based on rules and conditions.

- Able to spawn services and restart them on failures through signals and sockets.

- **Initialization Steps:**
  - Creates basic filesystem (/dev, /proc, /sys) and mounts it.
  - Parses **/init.rc**
  - Parses **/init.${hw_name}.rc** based on kernel command-line or /proc/cpuinfo
  - Build exec queues
  - Start triggers and associated actions and services.
    - e.g. "**early-init**", "**init**", "**early-fs**", "**fs**", "**post-fs**", "**early-boot**", "**boot**" …

Android Device Porting Walkthrough
Android Init Language

- **Actions:**
  - Named sequences of commands queued and executed upon events/triggers.
  - Syntax: **on <trigger> <command> <command> ...**
- **Triggers:**
  - Strings which can be used to match certain kinds of events and used to cause an action to occur.
- **Services:**
  - Programs which init launches and (optionally) restarts when they exit.
  - Syntax: **service <name> <pathname> [<argument>]* <option> <option> ...**
- **Options:**
  - Modifiers to services. They affect how and when init runs the service.
- **Commands:**
  - Android proprietary built-in commands.

Android Device Porting Walkthrough

Android Init Options

Option Syntax	Description
critical	This is a device-critical service. If it exits more than four times in four minutes, the device will reboot into recovery mode.
disabled	This service will not automatically start with its class. It must be explicitly started by name.
setenv <name> <value>	Set the environment variable <name> to <value> in the launched process.
socket <name> <type> <perm> [<user> [<group>]]	Create a unix domain socket named /dev/socket/<name> and pass its fd to the launched process.
group <groupname> [<groupname>]*	Change to groupname before exec'ing this service.
oneshot	Do not restart the service when it exits.
class <name>	Specify a class name for the service. All services in a named class may be started or stopped together
onrestart	Execute a Command when service restarts.
ioprio <rt|be|idle> <ioprio 0-7>	Set service priority and scheduling class.
console	Output logs on console.
capability <keycode>	Trigger service through keycode in /dev/keychord

Android Device Porting Walkthrough
Android Init Triggers

- **boot**
  - This is the first trigger that will occur when init starts (after **/init.rc** is loaded)
- **<name>=<value>**
  - Triggers of this form occur when the property <name> is set to the specific value <value>.
- **device-added-<path>**
- **device-removed-<path>**
  - Triggers of these forms occur when a device node is added or removed.
- **service-exited-<name>**
  - Triggers of this form occur when the specified service exits.

Android Device Porting Walkthrough

Android Init Commands

Command Syntax	Description
exec <path> [<argument>]*	Fork and execute a program (<path>). This will block until the program completes.
export <name> <value>	Set the environment variable <name> equal to <value> in the global environment.
ifup <interface>	Bring the network interface <interface> online.
hostname <name>	Set the host name.
chdir <directory>	Change working directory.
chmod <octal-mode> <path>	Change file access permissions.
chown <owner> <group> <path>	Change file owner and group.
chroot <directory>	Change process root directory.
class_start <serviceclass>	Start all services of the specified class if they are not already running.
class_stop <serviceclass>	Stop all services of the specified class if they are currently running.
class_reset <serviceclass>	Reset a class.
domainname <name>	Set the domain name.
insmod <path>	Install the module at <path>
mkdir <path> [mode] [owner] [group]	Create a directory at <path>, optionally with the given mode, owner, and group.
mount <type> <device> <dir> [<mountoption>]*	Attempt to mount the named device at the directory <dir>

Android Device Porting Walkthrough

Android Init Commands

Command Syntax	Description
setprop <name> <value>	Set system property <name> to <value>.
setrlimit <resource> <cur> <max>	Set the rlimit for a resource.
start <service>	Start a service running if it is not already running.
stop <service>	Stop a service from running if it is currently running.
restart <service>	Restart a service from running if it is currently running.
symlink <target> <path>	Create a symbolic link at <path> with the value <target>
sysclktz <mins_west_of_gmt>	Set the system clock base (0 if system clock ticks in GMT)
trigger <event>	Trigger an event. Used to queue an action from another action.
write <path> <string> [<string>]*	Open the file at <path> and write one or more strings to it.
rm <path>	Removes a file.
rmdir <path>	Removes a directory.
wait <file>	Wait for file to exist or timeout to be reached.
copy <src> <dest>	Copy from source to dest.
loglevel <level>	Set kernel log level.
load_persist_props	Load properties from files in /data/property

Android Device Porting Walkthrough
uEventd

- Somehow replaces udevd from desktop Linux.
- Used to set user/group/permissions on /dev nodes.
- **Steps:**
  - Parses **/ueventd.rc**
  - Parses **/ueventd.${hw_name}.rc** based on kernel command-line or /proc/cpuinfo.
  - Set nodes permissions.

Android Device Porting Walkthrough
Hardware Abstraction Library

- HAL Library for some HW Components (see **hardware/libhardware**):
  - **Audio** (Configuration, Mixing, Routing, Streaming, Echo Cancellation, FX ...).
  - **Framebuffer** (Configuration, Composition, Display).
  - **GFX Allocator** (GPU Memory Buffer Management).
  - **GFX HW Composer** (Surface Composition and Transformation).
  - **Camera** (Facing, Orientation, Buffer Management, Pictures, Recording ...).
  - **WiFi** (AP/STA/P2P Configuration and Firmware Support).
  - **GPS** (Configuration, Location Data Acquisition).
  - **Lights** (Backlight and LEDS control).
  - **Sensors** (Accelerometer/Pressure/Proximity/Gravity/... Controls).
  - **NFC**

Android Device Porting Walkthrough
Hardware Abstraction Library

- Series of dynamically loaded plugin (.so files):
  - MUST match board-specific plugin name
  - See APIs in **hardware/libhardware/include/hardware**
  - Default/dummy implementation provided in AOSP (**hardware/libhardware/modules**)
  - MUST be fully implemented by device vendor.
  - Can be closed-source.

- For each HAL class module plugins are checked:
  - based on system properties values in the following order:
    - ro.hardware
    - ro.product.board
    - ro.board.platform
    - ro.arch
  - in the following directories order:
    - /vendor/lib/hw
    - /system/lib/hw

Android Device Porting Walkthrough
LCD Display

- **Kernel Board Config:**
  - Disable HDMI device:
    - static struct omap_dss_device *omap4_panda_dss_devices[] = {
      - &omap4_panda_hdmi_device,
      + &omap4_panda_lcd_device,
      };
  - Use DPI-LCD as primary video device:
    - static struct omap_dss_board_info omap4_panda_dss_data = {
      .num_devices = ARRAY_SIZE(omap4_panda_dss_devices),
      .devices = omap4_panda_dss_devices,
      .default_device = &omap4_panda_lcd_device,
      };
  - Match custom DPI-LCD panel driver name:
    - static struct panel_generic_dpi_data omap4_lcd_panel = {
      .name = "my_dpi_lcd_panel",
      .platform_enable = omap4_panda_enable_lcd,
      .platform_disable = omap4_panda_disable_lcd,
      };
    - struct omap_dss_device omap4_panda_lcd_device = {
      .type = OMAP_DISPLAY_TYPE_DPI,
      .name = "lcd",
      .driver_name = "generic_dpi_panel",
      .data = &omap4_lcd_panel,
      .channel = OMAP_DSS_CHANNEL_LCD2,
      };

Android Device Porting Walkthrough
LCD Display

- **DPI-LCD panel driver config:**
  - Hack **drivers/video/omap2/displays/panel-generic-dpi.c**:

```
- static struct panel_config generic_dpi_panels[] = {
        [...]
        /* My custom DPI-LCD Panel */ {
                .x_res = 800,
                .y_res = 480,
                .pixel_clock = REFRESH_RATE * 1056 * 525 / 1000,
                [...],
                .config = OMAP_DSS_LCD_IVS | OMAP_DSS_LCD_IHS,
                .power_on_delay = 102,
                .power_off_delay = 100,
                .name = "my_dpi_lcd_panel",
        },
        [...]
 };
```

Android Device Porting Walkthrough
Orientation and Docked Tablet

- **Force Tablet Mode (120dpi)**
  - LCD 800x480 WVGA 7” (1.667 aspect ratio) => 133 dpi => Phone Mode.
  - In **BoardConfig.mk**:

```
PRODUCT_PROPERTY_OVERRIDES      += ro.sf.lcd_density=120
PRODUCT_CHARACTERISTICS          := tablet
```

- **Forced Landscape Orientation**
  - In **frameworks/base/core/res/res/values/config.xml**:

```
<bool name="config_allowAllRotations">false</bool>

<integer name="config_deskDockKeepsScreenOn">1</integer>

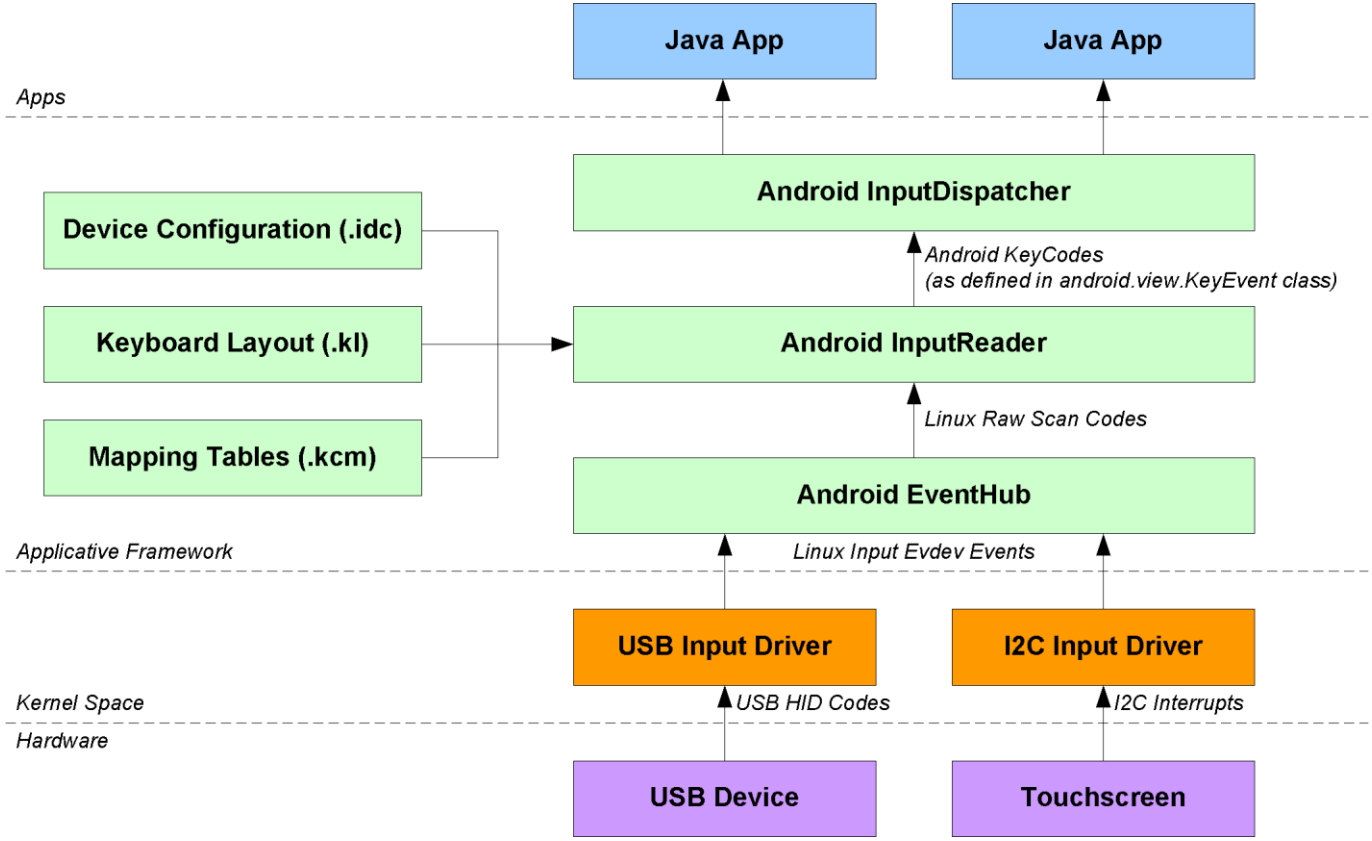
<bool name="config_deskDockEnablesAccelerometer">false</bool>
```

Android Device Porting Walkthrough
2D/3D GPU

- Force HW GPU Usage
  - In **BoardConfig.mk**:
    - BOARD_EGL_CFG := device/my_company/my_device/egl.cfg
    - BOARD_USES_HGL := true
    - BOARD_USES_OVERLAY := true
    - USE_OPENGL_RENDERER := true
- Deploy (in **/vendor**) vendor-provided binary blobs of:
  - Gralloc HAL Moduler
  - OpenGL|ES 1.x and 2.x libraries.
- Optionally force OpenGL|ES libs version detection.
  - In **BoardConfig.mk**:
    - PRODUCT_PROPERTY_OVERRIDES += ro.opengles.version=131072
    - => Makes AngryBirds happy ... and so are we !

Android Device Porting Walkthrough
Input Subsystem

- See http://source.android.com/tech/input/overview.html
- Many changes occurred with Android 4.0



Android Device Porting Walkthrough
Input Key Layout Files

- Maps Linux key and axis codes to Android key and axis codes.
- Required for all internal (built-in) input devices that have keys, including special keys such as volume, power and headset media keys.
- Location:
  - Located by USB vendor, product (and optionally version) id or by input device name.
  - The following paths are consulted in order:

/system/usr/keylayout/Vendor_XXXX_Product_XXXX_Version_XXXX.kl
/system/usr/keylayout/Vendor_XXXX_Product_XXXX.kl
/system/usr/keylayout/DEVICE_NAME.kl
/system/usr/keylayout/Generic.kl

/data/system/devices/keylayout/Vendor_XXXX_Product_XXXX_Version_XXXX.kl
/data/system/devices/keylayout/Vendor_XXXX_Product_XXXX.kl
/data/system/devices/keylayout/DEVICE_NAME.kl
/data/system/devices/keylayout/Generic.kl

Android Device Porting Walkthrough
Input Key Layout Files

- Syntax:
  - **key - Linux scan code number - Android key code name - [policy flags]**
  - key 1 ESCAPE
    key 114 VOLUME_DOWN WAKE
    key 16 Q VIRTUAL WAKE

- Policy flags:
  - **WAKE|WAKE_DROPPED**: The key should wake the device when it is asleep.
  - **SHIFT**: The key should be interpreted as if the SHIFT key were also pressed.
  - **CAPS_LOCK**: The key should be interpreted as if the CAPS LOCK key were also pressed.
  - **ALT**: The key should be interpreted as if the ALT key were also pressed.
  - **ALT_GR**: The key should be interpreted as if the RIGHT ALT key were also pressed.
  - **FUNCTION**: The key should be interpreted as if the FUNCTION key were also pressed.
  - **VIRTUAL**: The key is a virtual soft key that is adjacent to the main touch screen.

Android Device Porting Walkthrough
Input Device Configuration Files

- Contains device-specific configuration properties that affect the behavior of input devices.
- Optional for standard peripherals such as HID keyboard and mouse.
- Mandatory for built-in embedded devices such as touch screens.
- Location:
  - Located by USB vendor, product (and optionally version) id or by input device name.
  - The following paths are consulted in order:

/system/usr/idc/Vendor_XXXX_Product_XXXX_Version_XXXX.idc
/system/usr/idc/Vendor_XXXX_Product_XXXX.idc
/system/usr/idc/DEVICE_NAME.idc

/data/system/devices/idc/Vendor_XXXX_Product_XXXX_Version_XXXX.idc
/data/system/devices/idc/Vendor_XXXX_Product_XXXX.idc
/data/system/devices/idc/DEVICE_NAME.idc

Android Device Porting Walkthrough
I2C Touchscreen

- Driver needs to calculate correct position relative to screen.
  - Android does not do interpolation like X.Org xf86-input-evdev driver would do.

- **Kernel I2C Board Config:**
  - Pin Multiplexing:
    - /* TS IRQ - GPIO 48 */
      OMAP4_MUX(GPMC_A24, OMAP_MUX_MODE3 | OMAP_PIN_INPUT);
  - I2C Device Registration:
    - struct cy8ctmg110_pdata omap_panda_ts_data __initdata = {
      .irq_pin = GPIO_TS_IRQ,
      };

      static struct i2c_board_info __initdata panda_i2c_cy8ctmg110[] = {
      I2C_BOARD_INFO("cy8ctmg110", 0x20),
      .platform_data = &omap_panda_ts_data,
      };

      omap_register_i2c_bus(2, 100, panda_i2c_cy8ctmg110,
      ARRAY_SIZE(panda_i2c_cy8ctmg110));

Android Device Porting Walkthrough
I2C Touchscreen Driver

- **Initialization (driver name MUST match .idc filename):**
  - input_dev->name = "cy8ctmg110";

```
input_dev->evbit[0]                          = BIT(EV_SYN) | BIT(EV_KEY) | BIT(EV_ABS);
input_dev->keybit[BIT_WORD(BTN_TOUCH)]       = BIT_MASK(BTN_TOUCH);

input_dev->absbit[ABS_X]                     = BIT(ABS_X);
input_dev->absbit[ABS_Y]                     = BIT(ABS_Y);
input_dev->absbit[ABS_PRESSURE]              = BIT(ABS_PRESSURE);

input_set_abs_params(input_dev, ABS_X, min_x, max_x, 0, 0);
input_set_abs_params(input_dev, ABS_Y, min_y, max_y, 0, 0);
input_set_abs_params(input_dev, ABS_PRESSURE, 0, 1, 0, 0);
```

- **Touch Press Events:**
  -
```
static void ts_press_on(struct input_dev *input, int x, int y) {
        input_report_abs(input, ABS_X, x);
        input_report_abs(input, ABS_Y, y);
        input_report_abs(input, ABS_PRESSURE, 1);
        input_report_key(input, BTN_TOUCH, 1);
        input_sync(input);
}
```

- **Touch Release Events:**
  -
```
static void ts_press_off(struct input_dev *input, int x, int y) {
        input_report_abs(input, ABS_PRESSURE, 0);
        input_report_key(input, BTN_TOUCH, 0);
        input_sync(input);
}
```

Android Device Porting Walkthrough
Touchscreen Device Configuration File

- Add **cy8ctmg110.idc** to system:
  - In **device.mk**, add:
    - PRODUCT_COPY_FILES +=
      device/my_company/my_device/cy8ctmg110.idc:system/usr/idc/cy8ctmg110.idc

- Example of **cy8ctmg110.idc**:
  - # Basic Parameters
    touch.deviceType = touchScreen
    touch.orientationAware = 0

    # Size
    touch.size.calibration = none

    # Orientation
    touch.pressure.calibration = none

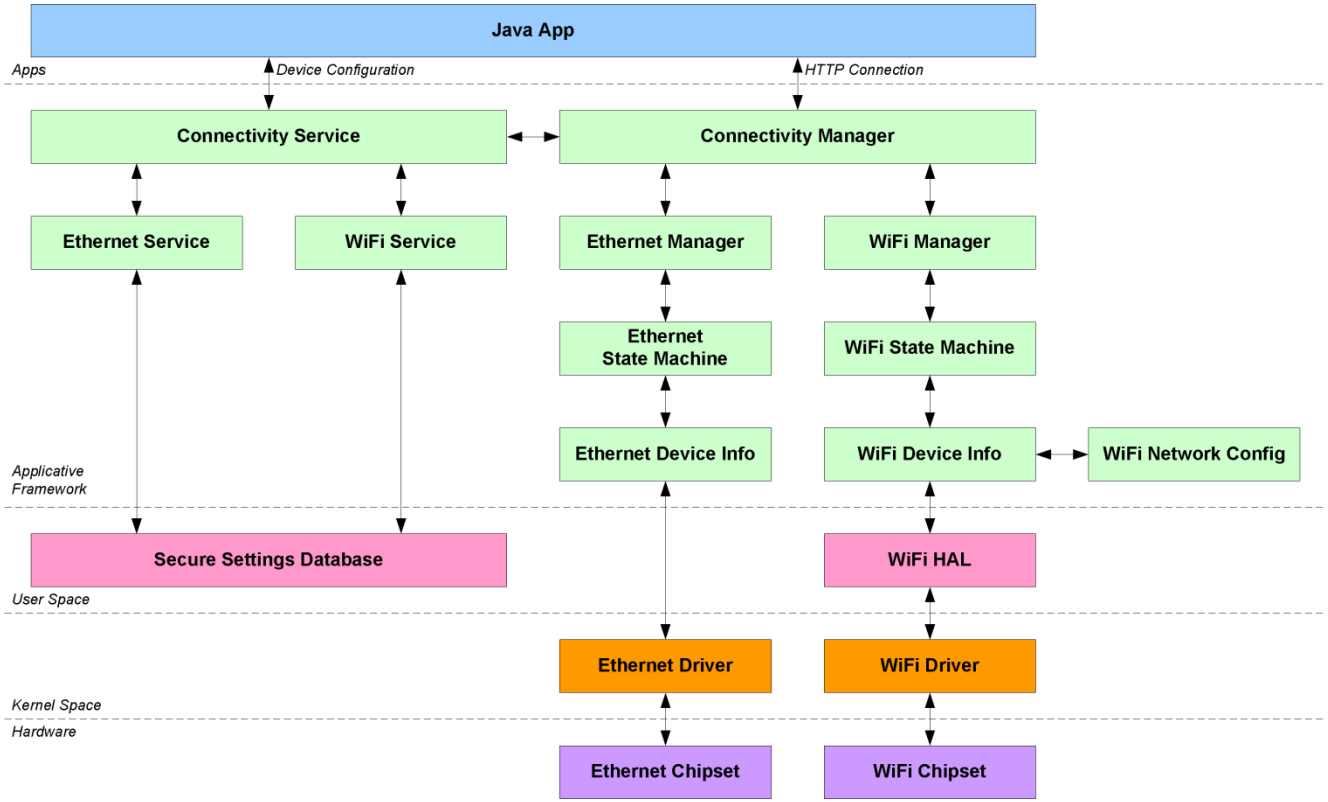
Android Device Porting Walkthrough
Storage Subsystem

- Overlay storage configuration file:
  - See **frameworks/base/core/res/res/xml/storage_list.xml**:
    - <storage
      android:mountPoint="/mnt/sdcard"
      android:storageDescription="@string/storage_internal"
      android:primary="true"
      android:removable="false"
      android:emulated="true"
      android:mtpReserve="100" />
- Edit list of mount points:
  - See **device/my_company/my_device/vold.fstab**:
    - dev_mount sdcard /mnt/sdcard 6
      /devices/platform/omap/omap_hsmmc.0/mmc_host/mmc0

Android Device Porting Walkthrough
Audio Subsystem

- In **BoardConfig.mk**:
  - BOARD_USES_GENERIC_AUDIO := false
    #BOARD_USES_ALSA_AUDIO := false
    BUILD_WITH_ALSA_UTILS := false
- MUST implement Audio HAL
  - See **hardware/libhardware/modules/audio** for example.
  - MUST declare as AUDIO_HARDWARE_MODULE_ID.
  - MUST implement audio HAL API; see **hardware/libhardware/include/hardware/audio.h**.
  - Implements audio in/out, volume get/set, mute get/set, gain get/set, list and configure all supported in/out devices, open/close input/output streams, configure mixer and router, add/remove audio effects (e.g. Automatic Gain Control and Acoustic Echo Cancellation).
  - MUST link against TinyALSA (new in ICS).
  - To be implemented in: **device/my_company/my_device/audio/audio_hw.c**
  - MUST be declared in LOCAL_MODULE as "audio.primary.${ro.product.board}"
    - e.g. : audio.primary.omap4
- See http://goo.gl/zPFQ8 for example.

Android Device Porting Walkthrough
Network & Connectivity



- Low-level kernel drivers work just like a charm up to Linux user-space.
- For Java apps connectivity, each connection type must register a specific ConnectivityManager and associated ConnectivityService that handles device configuration, packet routing and HTTP(S) proxy settings.

Android Device Porting Walkthrough
Network & Connectivity

- Overlay some resources in **frameworks/base/core/res/res/values/config.xml**:
  - <!-- Used by the connectivity manager to decide which networks can coexist -->

```
<string-array translatable="false" name="networkAttributes">
        <item>"wifi,1,1,1,-1,true"</item>
        <item>"bluetooth,7,7,0,-1,true"</item>
        <item>"ethernet,9,9,2,-1,true"</item>
</string-array>

<string-array translatable="false" name="radioAttributes">
        <item>"1,1"</item>
        <item>"7,1"</item>
        <item>"9,1"</item>
</string-array>
```

Android Device Porting Walkthrough
Ethernet Connectivity

- Ethernet connection type exists in ICS API.
- But with no ConnectivityManager or ConnectivityService implementation.
  - => Can't be configured nor used easily through regular Java apps (with exception to the Web Browser).
  - Experimental ECM patch has been integrated in Linaro ICS.
    - Allow low-level device configuration and settings.
    - Uses Persistent Secure Settings SQLite database.
    - Doesn't register with ConnectivityManager (i.e "No Internet Connection" symptom).
    - Doesn't provide HTTP(S) proxy support.
      - => To be implemented !! :-(

Android Device Porting Walkthrough
WiFi Connectivity

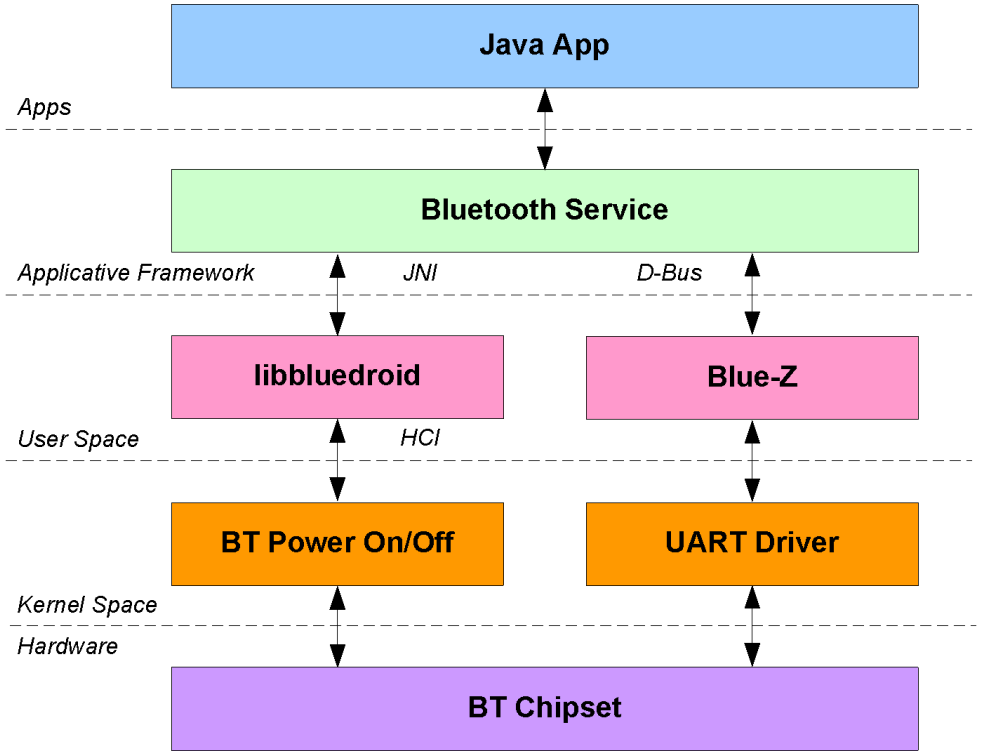
- Obviously requires kernel drivers + potentially associated firmware blobs.
- **Station Mode**:
  - In **BoardConfig.mk**:
    - WIFI_DRIVER_MODULE_PATH := "/system/modules/wl12xx_sdio.ko"
      WIFI_DRIVER_MODULE_NAME := "wl12xx_sdio"
      WIFI_DRIVER_FW_PATH_STA := "/system/etc/firmware/wl1271-fw.bin"
      WIFI_FIRMWARE_LOADER :=
    - Add WiFi hardware permission, in **device.mk**:
      - PRODUCT_COPY_FILES += frameworks/base/data/etc/android.hardware.wifi.xml:system/etc/permissions/android.hardware.wifi.xml
  - HAL driver in **hardware/libhardware_legacy/wifi**.
    - Default implementation should be sufficient in most cases.
    - Loads/unloads kernel drivers, loads up firmware and register UNIX socket connection to WPA supplicant for further control.
    - Used by JNI Java framework.

Android Device Porting Walkthrough
WiFi Connectivity

- **Access Point Mode**:
  - In **BoardConfig.mk**:
    - WIFI_DRIVER_FW_PATH_AP := "/system/etc/firmware/wl1271-fw-ap.bin"
      BOARD_WPA_SUPPLICANT_DRIVER := WEXT
      WPA_SUPPLICANT_VERSION := VER_0_8_X
      BOARD_WPA_SUPPLICANT_PRIVATE_LIB := private_lib_driver_cmd
    - See **external/wpa_supplicant_8/private_lib**.
    - In **init.rc**:
      - service wpa_supplicant /system/bin/wpa_supplicant -Dwext -iwlan0 –dd
        socket wpa_wlan0 dgram 660 wifi wifi
        disabled
        oneshot
  - In **wpa_supplicant.conf** (both for AP and STA):
    - ctrl_interface=DIR=/data/system/wpa_supplicant GROUP=wifi
      update_config=1
      ap_scan=1

Android Device Porting Walkthrough
Bluetooth

- Obviously requires kernel drivers + potentially associated firmware blobs.
- In **BoardConfig.mk**:
  - BOARD_HAVE_BLUETOOTH := true
- Add Bluetooth hardware permission, in /system/etc/permissions/
  - <feature name="android.hardware.bluetooth" />
- May requires support in audio HAL for BT A2DP support.
- **libbluedroid** implements enables/disables BT interface and creates HCI socket through rfkill (see **system/bluetooth/bluedroid**).



Android Device Porting Walkthrough
Multimedia Subsystem

- In **media_profiles.xml**:
  - XML file that lists audio/video/image encoder/decoder capabilities.
  - <!-- If a codec is not enabled, it is invisible to the applications. In other words, the applications won't be able to use the codec or query the capabilities of the codec at all if it is disabled -->
    <VideoEncoderCap name="h264" enabled="true" minBitRate="64000" maxBitRate="3000000" minFrameWidth="176" maxFrameWidth="800" minFrameHeight="144" maxFrameHeight="480" minFrameRate="1" maxFrameRate="30" />
  - No checks on decoder capabilities so far.
- OpenMAX components enablement in **BoardConfig.mk**:
  - ```
    HARDWARE_OMX                    := true
    ifdef HARDWARE_OMX
        OMX_VENDOR                  := ti
        OMX_VENDOR_WRAPPER          := TI_OMX_Wrapper
        BOARD_OPENCORE_LIBRARIES    := libOMX_Core
        BOARD_OPENCORE_FLAGS        := -DHARDWARE_OMX=1
    endif
    ```
  - OMX implementation and StageFright integration really comes as a dedicated talk.
    ```

Android Device Porting Walkthrough
No-Battery Trick

- Partial trick for Android to believe it's running on power supply (at least on main UI).
  - Add new system setting in **device.mk**:
    - PRODUCT_PROPERTY_OVERRIDES += hw.nobattery=true
  - Hack on **frameworks/base/services/java/com/android/server/BatteryService.java**
    - String hwNoBatteryStr = SystemProperties.get("hw.nobattery");
      boolean hwNoBattery = Boolean.parseBoolean(hwNoBatteryStr);

```
if (!hwNoBattery)
        mPowerSupplyObserver.startObserving("SUBSYSTEM=power_supply");

private void stubUpdate() {
        // Hardcode values. We could read them from properties
        mAcOnline                               = true;
        mUsbOnline                              = false;
        mBatteryPresent                         = true;
        mBatteryLevel                           = 100;
        mBatteryVoltage                         = 4700;
        mBatteryTemperature                     = 80;
        mBatteryStatus                          = BatteryManager.BATTERY_STATUS_FULL;
        mPlugType                               = BatteryManager.BATTERY_PLUGGED_AC;
}

private synchronized final void update() {
        if (hwNoBattery)
                        stubUpdate();
        else {
                        native_update();
                        processValues();
        }
}
```

Android Device Porting Walkthrough
Phone

- Overload some resources in **frameworks/base/core/res/res/values/config.xml**:

```
- <!-- This device is not "voice capable"; it's data-only. -->
  <bool name="config_voice_capable">false</bool>

  <!-- This device does not allow sms service. -->
  <bool name="config_sms_capable">false</bool>

  <!-- Enables SIP on any network interface -->
  <bool name="config_sip_wifi_only">false</bool>

  <!-- Enables built-in SIP phone capability -->
  <bool name="config_built_in_sip_phone">true</bool>
```

Android Device Porting Walkthrough
What's More ?

- More devices to be supported and described …
  - Modem / RIL
  - Camera
  - GPS
  - Sensors (Accelerometer …)
  - HW OpenMAX Multimedia Encoders/Decoders
- But that's really another story ... for another day ;-)

Thank You



AT
THE
SPEED
OF
IDEAS™

Alcatel·Lucent

www.alcatel-lucent.com