What else can you do with Android?

Porting to custom hardware

Chris Simmonds, 2net Limited

Class TU-3.2

Overview

- Porting Android to custom hardware
  - Creating an Android kernel
  - Building the Android Open Source
  - Device support: input, display, network
  - Initialising services
  - Configuring ...

Steps to implement Android

- Get a kernel with Android patches
- Build the Android user space
- Customise
  - Boot sequence: services
  - Networking: Ethernet
  - Input devices: touchscreen & user buttons
- Write application – in Java
  - Probably have to write some native (C/C++) code

Android Open Source Project

- Aka AOSP. Home page:
  - http://source.android.com
- A multitude of git trees:
  - http://android.git.kernel.org
- Use repo tool and a manifest file to download and sync with a sub-set of the trees
- Note: most development is done privately and then released to the AOSP later
  - AOSP trees lag the code available to OHA members

Chip support

- armv5te and armv7a out-of-the-box
- MIPS, PPC, x86, SH ports have been done
- SoC support (kernel and drivers) from vendors including
  - Freescale i.MX, TI OMAP
- Custom board level support
  - In most cases you will have to create your own Android BSP
  - Requires merging Android kernel patches into your own kernel tree

Android kernel repository

- http://android.git.kernel.org/ is the main repository for the AOSP
- There are several kernel trees there: the main one is kernel/common.git. It has several heads
  - android-2.6.35
  - android-2.6.32
  - android-goldfish-2.6.29
  - android-2.6.29
  - android-goldfish-2.6.27
  - android-2.6.27
  - android-2.6.2

Android kernel additions

- Wakelocks & Power management
- Binder
  - corba-like IPC, replaces SYSV and POSIX IPC
- ashmem
  - Android shared memory: replaces SYSV & POSIX shared mem
  - Apps use ioctl and mmap on /dev/ashmem
- pmem
  - Process memory allocator
- logger
  - Ring buffers for logging: used by logcat application
- oom handling
  - "better" oom hander
- Other bibs and bobs...

Notes about merging Android kernel

- Android patches not in mainline
  - Some changes were in drivers/staging/android for kernels up to 2.6.32
- Merging - the big picture
  - clone Android kernel
  - create a patch between Android and mainline
  - apply the patch to your kernel & resolve the errors
- Details...
  - read section 4 in the handout

AOSP: repo

- Everything except the kernel is available in the AOSP
- repo is a tool to synchronise the git trees you need
  - Obtain from http://android.git.kernel.org/repo
  - Initialise the repository
  - The trees to clone are defined in a manifest
    - an xml file

Example of using repo

- Using the default manifest
  - note: takes a long time & downloads several GiB of code

```
mkdir ~/myandroid
cd myandroid
repo init -u git://android.git.kernel.org/platform/manifest.git
repo sync
```

Building everything

- Next, build it like this
  - the final 'm' is an alias for 'make' (!)
  - this also takes a long time and consumes much disk space

```
cd myandroid
export JAVA_HOME=$HOME/jdk1.5.0_22
PATH=$JAVA_HOME/bin:$PATH
. build/envsetup.sh
m
```

What you get

- Tools (e.g. adb ,emulator) in
  - myandroid/out/host/linux-x86
- Target file images (system.img, userdata.img, ramdisk.img) in
  - myandroid/out/target/product/generic
- Individual target run-time files in
  - myandroid/out/target/product/generic/root/

Building the sdk

- Build your own sdk with

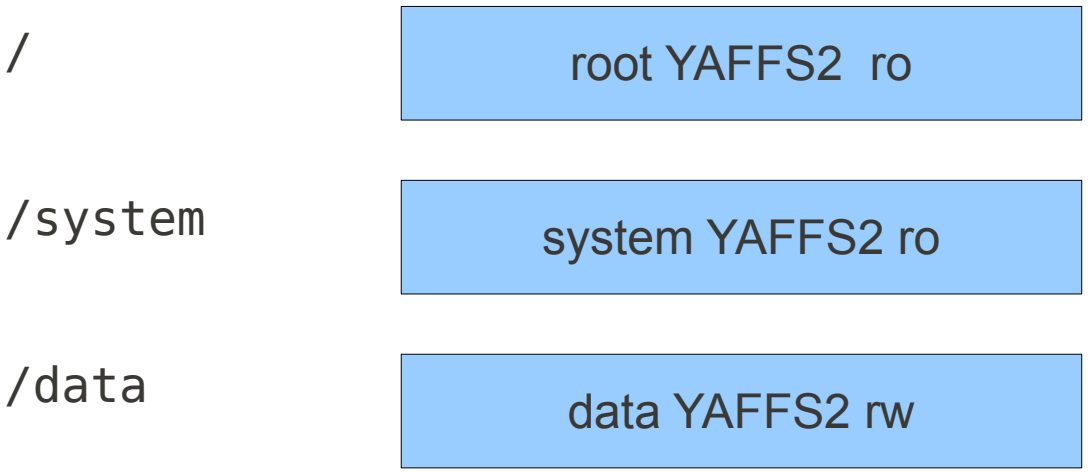
```
m sdk
```

- The output is in

```
myandroid/out/host/linux-x86/android-
sdk_eng.<user name>_linux-x86/
```

Default file systems

- This is the default layout in NAND flash memory
  - using the YAFFS2 file system format
- Only /data is writeable: this is where your apps and settings are stored



More about booting: init.rc

- The first program to run is **/init**
- Creates device nodes, starts services
  - A bit like a combination of ordinary init and udev
- Controlled by script in **/init.rc:**

```
on init
sysclktz 0
loglevel 3

# setup the global environment
    export PATH
/sbin:/system/sbin:/system/bin:/system/xbin
    export LD_LIBRARY_PATH /system/lib

...
```

Android init language: actions

An action is what to do when a trigger is received.
Actions take the form of:

```
on <trigger>
   <command>
   <command>
   <command>
```

For example the init trigger is issued when the system starts, so init.rc begins with this action

```
on init
```

Android init language: services

An service is usually a daemon process that is started
Services take the form of:

```
service <name> <pathname> [ <argument> ]*
   <option>
   <option>
```

The service class is set by class <class_name>. If there is no class it defaults to 'default'.

You start services of a certain class with class_start, so this command at the end of the on init action starts all the default services

```
class_start default
```

Example service in init.rc

```
## Daemon processes to be run by init.
##
service console /system/bin/sh
    console

# adbd is controlled by the persist.service.adb.enable system property
service adbd /sbin/adbd
    disabled

# adbd on at boot in emulator
on property:ro.kernel.qemu=1
    start adbd

on property:persist.service.adb.enable=1
    start adbd

on property:persist.service.adb.enable=0
    stop adbd
```

adb is disabled by default

enabled if running emulator
or persist.service.adb.enable=1
is set in /default.prop

Display devices

- Uses Linux frame buffer device
- Screen size limited in framework classes
  - 1024x768...
  - Gingerbread rumoured to support 720p HD (1280x760)
- 2D acceleration (optional)
  - Custom library 'SGL'
- 3D acceleration (optional)
  - OpenGL ES 1.0

Input devices - buttons

- Uses Linux input device framework interface (/dev/input/event*)
  - So you need to write a normal Linux driver that creates input events
- The default key mapping is in

  `/system/usr/keylayout/qwerty.kl`
- A key mapping may be qualified with WAKE
  - Pressing this key while the device is asleep will wake it up

Input devices - touchscreen

- Also uses Linux input device framework
  - So, any Linux touch screen driver will work
  - Note: does not use tslib. Filtering done in Android framework
- Input classes smart enough to display a soft keyboard when needed if there is no real keyboard
- If you want screen, use ts_calibrator
  - Source is in the AOSP tree

Summary

- There is quite a lot to know if you are porting Android to custom hardware
  - But not more than other embedded Linux distros
- All of the source is open and easily available
- The sort of customisations you can make without much effort include
  - changing the boot-up script, init.rc
  - adding drivers for input devices
  - adding drivers for accelerated graphics

Links

- Relevant Android developer forums
  - http://groups.google.com/group/android-porting
  - http://groups.google.com/group/android-kernel
- Inner Penguin blog at
  - http://www.embedded-linux.co.uk
- 2net web site
  - http://www.2net.co.uk