Using Chroot to Bring Linux Applications to Android

Getting the best of both worlds…

Mike Anderson
Chief Scientist
The PTR Group, Inc.
mike@theptrgroup.com
What We’ll Talk About…

- Why mix Android and Linux?
- Android under Linux
- Linux under Android
- Communicating between the domains
What are we trying to Accomplish?

Android is probably the most widely deployed version of Linux on the planet
  - We want to extend the platform to handle other tasks without extensive modification of the underlying framework

Enable porting of Linux applications to Android

Ease package management issues by allowing easy access to Linux repositories

Get an optimal mix of Linux and Android for use in non-phone applications
Advantages of Android

- Tremendous market position
- Well-defined development and deployment environments
  - Great application framework with good modularity
    - Network, audio, power, etc.
- Well-understood GUI/UX
- Good selection of Java libraries
- Availability of NDK gives option for higher performance than Java implementation

Source: pctechmag.com
Good Integration of SDK

```java
package project.four.adk;

import java.io.FileDescriptor;

public class ProjectFourActivity extends Activity {
    private static final String TAG = ProjectFourActivity.class.get
    private PendingIntent mPermissionIntent;
    private static final String ACTION_USB_PERMISSION = "c" + "om.android.example.USB_PERMISSION";
    private boolean mPermissionRequestPending;
    private UsbManager mUsbManager;
    private UsbAccessory mAccessory;
    private ParcelFileDescriptor mFileDescriptor;
}
```
Disadvantages of Android

- Package management
  - Difficult to update the underlying framework

- Library and application availability
  - Purpose-built for phones/tablets and not much else

- Extensions to elements like lib sensors requires rebuilding the AOSP sources

- GUI choice dictates the kernel choice
  - 4.1 is different from 4.4
    - Look and feel are different too
  - Difficult to go off the path set by Google

- Android SCM does not facilitate easy extensions by non-OHA folks
Bionic libc Compatibility Issues

- Restricted POSIX compatibility
- No C++ exceptions
- No locales or wide char support
- Several missing functions like `getpwd()`
- Really built as a single-user user space
- More info found in `bionic/libc/CAVEATS`
- These issues and more make it difficult to port standard Linux applications to Android

Source: slideshare.net
Different Views of the World

Linux and Android see things differently

Linux

- Bootloader
- Linux Kernel
- OS Libraries
- OS utilities, runtime, etc.
- Apps

Android

- Bootloader
- Linux Kernel
- Init
- HALs, flingers, etc.
- Application Framework
- Applications

ABS- SClara- 0428- 8
Copyright 2014, The PTR Group, Inc.
The Ideal World

In the ideal world, we could just use the Android framework and get the UX

Unfortunately, Android is a tightly-coupled architecture that makes that very difficult

These elements need to be kept intact for Android to function
Several Approaches…

If we want to run Linux code under Android, we could:

1. Port the Linux code to bionic libc
   - Problematic due to differences between bionic and glibc
2. Run Android as a package under Linux
   - The approach taken by Pragmatux
3. Run Linux applications in a chroot environment
4. Extend LXC to support options 2 or 3 better
What is “chroot”? 

Chroot is a command that was introduced into Unix in 1979.

Changes the apparent root file system for the calling process and its children.

- Used for development and testing when the target O/S release is different from the development host.

Once running in chroot, applications can’t typically get to files outside of the chroot.

- Often known as “chroot jail”

Only root user can execute the chroot command.

Source: bukisa.com
Pragmatux

Found at http://www.pragmatux.net/

- Project leads are Bill Gatliff and Ryan Kuester

Hardware boots Linux

- Uses a Debian-like approach for repos

Leverages idea that Android file system has little overlap with Linux file system

- /proc, /etc, /dev, /sys are a few exceptions
  - Uses bind mounts to keep things straight

Primary goal is to use Android framework for the UI but keep predictability of Linux for embedded applications
Android File System in chroot

- We can encapsulate the Android environment into the embedded Linux file system.

- Sockets and kernel communications work as normal.

Source: insymbols.com
+ Linux is in charge and we can use modern kernel with PREEMPT_RT and Android code from staging tree for Android support
+ Gives us Android UX with HRT/SRT support for control applications
  - We can prioritize Android apps as needed
+ Helps keep costs down
  - Only one CPU needed
    - Multi-core is a big plus

- We need to tweak the Android init sequence so Android doesn’t take over the device
  - We need to do the bind mounts as well
- Complexity can be troublesome
- We need enough RAM and storage for 2 O/S user spaces
An alternate approach is to host the Linux file system in the Android F/S.
The Android device must be rooted for this approach to work.
Using **chroot**, we can create an alternate root file system that Linux applications can live in easily.

Linux can live with `/bin`, `/etc`, `/dev`, `/lib`.

- `/proc` and `/sys` can be bind mounted.

Alternatively, we can loop mount an image and chroot to the mounted image.
- Gives us a full Linux in our Android.
+/

+ 
- There are already apps on Google Play that streamline this sort of installation 
- We get Linux package management capabilities 
- You can use VNC to get GUI-centric Linux applications running 
- Only one CPU needed 
  - Multi-core is a big plus 
- We only need to install the libraries and minimum files to run 

- 
- RAM and storage requirements vary depending on applications being run 
- Android framework is in charge 
- Not likely that the kernel will have PREEMPT_RT or other latency settings 
- Development environment can be tricky 
  - It’s possible to install development environment on Android platform in chroot
Simple Example

We had a customer that needed to run some RedHat-based programs, but wanted to get the Android UX

- Media-scanning kiosk device
  - Looking for malware on user media
- Cut down on training time for users and get touch-screen support
- Developers were mostly Java centric

We constructed an Android x86 platform running Atom using a COTS motherboard

Built Android from AOSP sources and edited libsensors for the devices we had on the motherboard
Simple Example #2

Using `ldd` we were able to isolate the application and required libraries to the bare minimum

- Installed chroot was < 100 MBs with the app and libraries

We created a daemon that ran in the chroot that listened for requests from the Android app via socket communications

We then created the Android application that passed configuration and scanning requests to the daemon that dispatched the application and returned responses to Android
Simple Example #3

- Small team of 2.5 FTEs to build Android, chroot components and interface daemon
  - We had to start the chroot and the daemon from the Android init process

- Two month project including custom enclosure

- Final product:
Step-by-step for an Android Device

- Make sure you have the device rooted
- Go to the Google Play Store
  - Install busybox, terminal emulator and VNC client if you need Linux window manager
  - Search for “Linux Install” and you should find several apps that can install Linux
    - Pick one and install it
- Start the Linux installer, pick your distribution and download it
  - Follow the steps to install it
- Voila! Linux on your Android device
  - Linux will see your Android devices /dev and the network will just work
  - You can start an ssh server, VNC server, web server, etc. automagically
Tuning the Linux Side

Linux will be a console application visible in the terminal emulator

- Graphical Linux applications will use VNC for display

Use the Linux package manager for your variety to install additional package as needed

You will need to edit the ~/.vnc/xstartup to add the applications you want to start on VNC connection

- I installed lxde, but others are possible

Set your VNC password using vncpasswd

Start your VNC client for window manager goodness 😊
Example Walkthrough

Complete Linux Installer

What's new

v2.7.2
* Removed Location permissions, these where required by ad network but have disabled this option, READ_PHONE_STATE is still needed
* Added indiegogo campaign to splash screen, please see http://www.indiegogo.com/projects/linuxonandroid/x/111384 for more info and help support the project

Description

We are currently running a indiegogo campaign to take the project to the next level! please http://www.indiegogo.com/projects/linuxonandroid/x/111384 for more info

Complete your installation and you'll have a first entry to installing Linux. Setup on your

First Launch

First Start will now install the bootscripts and busybox needed for Linux booting, this will not harm your device but does require root!

OK
Example Walkthrough #2
Example Walkthrough #3

Install Debian

Let's start by downloading a Debian image file.

Download Image

Now while this is downloading, download and install the terminal app and vncviewer app if you have not already

Download/Install VNCViewer App

Download/Install Terminal App

Once you have downloaded the Linux package and installed both apps go to the next tab.

Install Guides

Android version: 4.3 CPU type: ARM
These are the Linux distros that support your device.

- Ubuntu 13.04
- Ubuntu 13.10
- Debian
- Debian Testing
- ArchLinux
- Kali Linux
- Fedora 19
- openSUSE 12.3

Copyright 2014, The PTR Group, Inc.
Launch Linux...

Please select Linux distribution below to start it or change settings.

Debian

Start Linux

This is the one click solution for booting into Linux.

Press the menu button on your device to edit the presets to match where you have set.

You can then either launch any of the linux distros from here or place a widget on your home screen.

This new and highly improved one click boot system was designed and coded by Mark. 

Example Walkthrough #4
Example Walkthrough #5

Launch VNC

ADDRESS
127.0.0.1

NAME
Debian

DETAILS
Signature
Username
Password
Communications Between Domains

- Android is missing most of the POSIX IPC mechanisms
  - No message queues, shared semaphores, etc.
- IP sockets work fine
  - Path of least resistance
- You can create your own communications channels via the kernel
  - Device drivers work via the kernel
- `/proc` and `/sys` work too
Summary

Via the common Linux kernel, it is possible to co-host Linux and Android apps at the same time

- Easier than porting the application to Android
- Allows you to extend Android with existing Linux ARM repos
- Gains access to a package management system that’s more flexible than Google Play Store
- Allows you to do native ARM development on Android

Multi-core platforms with at least 2 GBs of RAM work reasonably well

You can tune the chroot to contain just what you need

- Smaller footprint

It’s fun to have everything at your finger tips in one portable platform