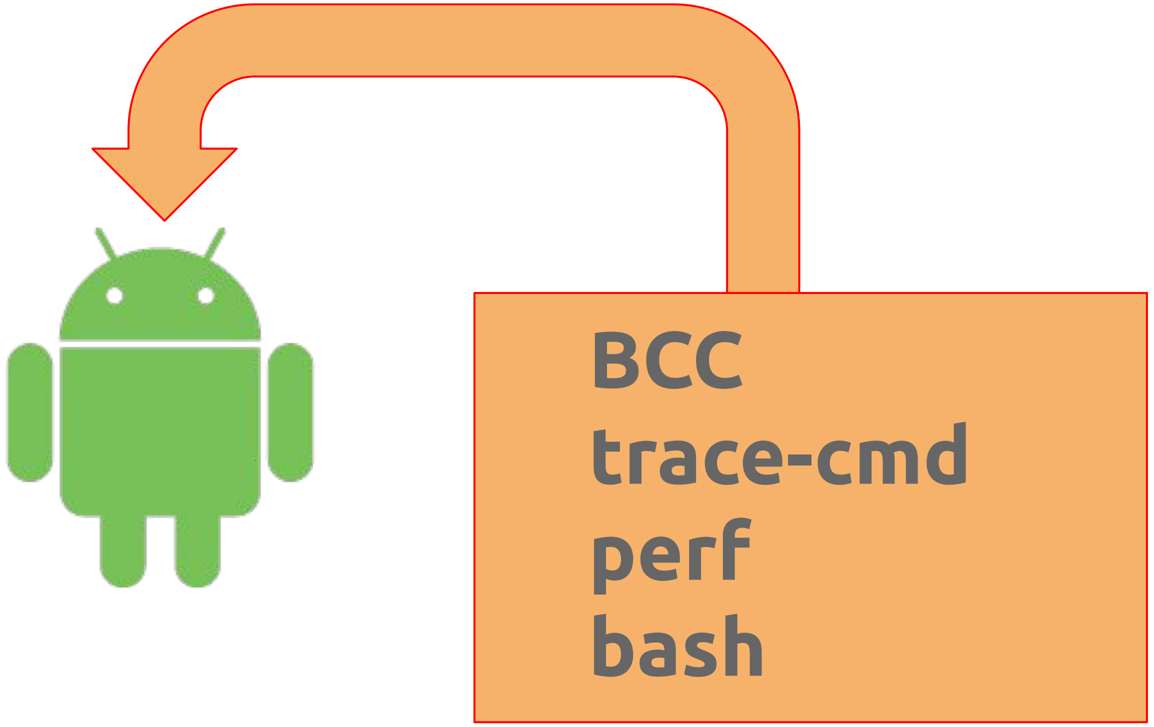
adeb:
The better adb shell

A chroot-based "adb shell" for Android

Joel Fernandes <joelaf@google.com>
www.joelfernandes.org

Google

My usecase



Broadly speaking...

- **Run** ANY open source package on Android device (any arch but ARM for now).

- Either **binary** form or native-build it from **source**.

Arm on Arm
Building software on Arm
Linaro

Typically what people do...

- **Cross-compile** and push **static** binary
  - Error prone
  - Crippled
  - Limited

- Need a better way...

Android Userspace is awesome, but...

- Designed for Android framework
- Android Build system can be a pain
- Licensing issues

Problems with cross-compiling things

- Many open source packages **refuse to cross compile**
- **Slower** develop-test-develop cycle
- Tools like BCC are **difficult** (impossible) to get working

Solution in a nutshell

- **Build** rootfs using qemu-debootstrap
- **Push** a prebuilt (or build one) **root fs** to /data
- **Run** adb shell with **chroot(2)** of /data/.../bash

In reality several other things happen:

- Setting up mounts correctly
- Setting up /etc/passwd so networking works
- Setting up kernel headers
- Setting up tty and bash environment etc.

Trying to solve fragmentation of chroot

- Everyone does their own chroot for Android
  - Duplicated effort
  - New users don’t know how to do it properly

- Let’s unify our efforts and use adeb…

Demo : Compiling rt-app

Demo : Compiling perf   (8 cores.. 37 seconds!)

Demo : Run a rust program

Demo : disassemble android binaries

Demo : Compile kernel (8 cores.. 15m 37s)

Demo : Prepare...

```
adeb prepare
adeb prepare --full
```

Demos of BCC tools on Android

runqlen: Per-CPU Histogram of run queue lengths

```
taskset -a -c 6 hackbench -P -g 2 -f 2 -l 10000000 &

# runqlen -C
cpu = 4
     runqlen       : count     distribution
        0          : 68       |****************************************|

cpu = 5
     runqlen       : count     distribution
        0          : 49       |****************************************|

cpu = 6
     runqlen       : count     distribution
        0          : 0        |                                        |
        1          : 79       |*********************                   |
        2          : 10       |**                                      |
        3          : 81       |*********************                   |
        4          : 149      |****************************************|
```

BCC “trace” running in adeb : A swiss army knife

Usecase: Using dynamic tracepoints (kprobes)

```
Function we’d like to trace has prototype:
long do_sys_open(int dfd, const char __user *filename, int flags, umode_t mode);

# trace 'do_sys_open "%s", arg2' -T

TIME     PID     TID     COMM            FUNC             -
19:45:44 2220    2250    storaged        do_sys_open      /sys/block/sda/stat
19:45:44 2220    2250    storaged        do_sys_open      /sys/block/sda/stat
19:45:48 2132    2132    servicemanager  do_sys_open      /proc/4113/attr/current
19:45:49 2352    2437    DeviceStorageMo do_sys_open      /system/framework/arm/boot.art
19:45:49 2352    2437    DeviceStorageMo do_sys_open      ../system@framework@boot.art
19:45:49 2352    2437    DeviceStorageMo do_sys_open      /system/framework/arm64/boot.art
19:45:49 2352    2437    DeviceStorageMo do_sys_open      ../system@framework@boot.art
19:45:55 2132    2132    servicemanager  do_sys_open      /proc/2480/attr/current
19:45:55 2132    2132    servicemanager  do_sys_open      /proc/2480/attr/current
```

Resources

- adeb or Androdeb: https://tinyurl.com/androdeb

Questions or Comments?