TOWARDS BUILDING MODERN ANDROID AS A GUIX PACKAGE

Julien Lepiller
Guix 10th anniversary
September 2022
Introduction

Android System
Introduction

Android System

Application
Introduction

Android System

Application

Dependencies

`androidx.appcompat:appcompat`
Introduction

Android System

Application

Dependencies

Gradle

```
androidx.appcompat:appcompat
```
Introduction

Android System

Application

Dependencies

Gradle

Kotlin

androidx.appcompat:appcompat
Introduction

Android System

Application → Dependencies

Gradle → Kotlin

androidx.appcompat:appcompat
Introduction

Android System

Application

Dependencies

Gradle

Kotlin

androidx.appcompat:appcompat

Host Tools

SDK
Introduction

Android System

Application

Dependencies

SDK

Host Tools

Gradle

Kotlin

androidx.appcompat:appcompat
Agenda

- **Getting Sources**
- Analyzing Blueprint Files
- Creating a Build System
- Current Status
- What’s Next?
Official instructions (simplified):

```bash
# Get sources
repo init -u https://android.googlesource.com/platform/manifest
call tool repo sync
# then build android
source build/envsetup.sh
lunch aosp_arm-eng
m
```
The Manifest

...<project path="device/ti/beagle_x15-kernel"
    name="device/ti/beagle-x15-kernel"
    groups="device,beagle_x15,pdk"
    clone-depth="1" />
<project path="system/libziparchive"
    name="platform/system/libziparchive"
    groups="pdk" />
<project path="prebuilts/abi-dumps/ndk"
    name="platform/prebuilts/abi-dumps/ndk"
    groups="pdk-fs" clone-depth="1" />
...
Manifest Parsing

(use-modules (sxml simple))

- Shallow clone of each project
- Exclude prebuilts
28 GB later...
Repository Hierarchy

- Lots of repositories
- Each may contain multiple projects / packages / targets
- Vendored dependencies
- Different types: host, target, C, java, etc.
## SDK Build System: Blueprint and Soong

### Blueprint
- Meta build system
- Defines syntax, basic rule handling with defaults
- Defines basic rules for Go, so it can build itself
- Nice bootstrap included :)

### Soong
- Build system for Android
- Based on blueprint
- Defines additional rule types for Android
  - Java
  - C/C++
  - Go
  - Scripting
  - etc.
Soong failures (for Guix purposes)

- Refuses to run arbitrary binary for reproducibility reasons
- Runs blessed prebuilt binaries for “reproducibility”
- Assumes it is in a directory with all of Android sources
  - Only looks for dependencies in intermediate result directory
  - Install to intermediate result directory
- hard-coded targets in the build directory
Agenda

- Getting Sources
- Analyzing Blueprint Files
- Creating a Build System
- Current Status
- What’s Next?
The goal is to build the SDK host tools (adb, etc.). They use Blueprint:

```plaintext
cc_binary_host {
  name: "adb",
  stl: "libc++_static",
  defaults: ["adb_defaults"],
  srcs: [
    "client/adb_client.cpp",
    "client/bugreport.cpp",
    ...
  ]
}
```
PEG Parser


```
(use-modules (ice-9 peg))

(define-peg-pattern keyval all
  (and var-name (* SP) COLON (* SP) keyval-val))
(define-peg-pattern var-name body
  (+ (or "_" (range \a \z) (range \A \Z)
      (range \\0 \9))))
(define-peg-pattern keyval-val body
  (or boolean integer strings-list string-all map-pat var-name))
```
Transform to a Guile Structure

cc_library {
  name: "foo"
  srcs: ["foo.c"]
}

cc_binary {
  name: "bar"
  srcs: ["bar.cc"]
  shared_libs: ["foo"]
}
(define-module-record-type cc make-cc cc?
    (srcs cc-scrs (default '()))
    (shared-libs cc-shared-libs (default '()))
    ...
)

- Structure for each family of types
- Defines all possible keys
- Fill in keys from defaults, then current module
Agenda

- Getting Sources
- Analyzing Blueprint Files
- **Creating a Build System**
- Current Status
- What’s Next?
Generate abstract *Makefile* rules from the specified blueprint module

- **out** files created
- **deps** files required
- **cmd** command to generate output files
- **source?** whether it generates sources
- **dependencies** external dependencies (for importer)

Run `make`, `make install`.
Phases are run on the build side

- All go to (android build) separate modules
- (android build soong-build-system): phases top-level
- (android build blueprint): blueprint parsing
- (android build soong cc): CC and Art targets specific code
- (android build soong java): Java targets specific code
(use-modules ((guix build gnu-build-system) #:prefix gnu:))
(define %standard-phases
  (modify-phases gnu:%standard-phases
      (delete 'bootstrap)
      (add-before 'configure 'reset-include-path reset-include-path)
      (replace 'configure configure)
      (delete 'check)))
Show me the code

Let’s look at the code of (android build-system soong)
Agenda

- Getting Sources
- Analyzing Blueprint Files
- Creating a Build System
- **Current Status**
- What’s Next?
Guix Android Channel

(cons* (channel
    (name 'guix-android)
    (url "https://framagit.org/tyreunom/guix-android.git")
    (introduction
      (make-channel-introduction
        "d031d039b1e5473b030fa0f272f693b469d0ac0e"
        (openpgp-fingerprint
          "1EFB 0909 1F17 D28C CBF9 B13A 53D4 57B2 D636 EE82"))))
%default-channels)
use (android import repo)
(define manifest (get-manifest))
(define root (fetch-manifest-repositories manifest)); 28GB
(define bp-files (get-all-bp-files root)); find-files, ~1-2 minutes
(define bp-maps (get-bp-maps bp-files)); ~5 minutes on HDD
(import-recursively manifest root bp-maps '("adb" "fastboot")
"sources.scm" "packages.scm")
Packages and Experiments

- aapt, aapt2, adb, aidl, apksigner, dexdump, dmtracedump, dx, etc1tool, fastboot, hprof-conv, libaapt2_jni, sqlite3, split-select, zipalign
- Soong build system for cc, genrule, java and protobuf
- Cross-compilation ready
- Experimenting a Kotlin bootstrap (failure after 20-ish versions)
- Sbt bootstrap (from Scala binary)
Agenda

- Getting Sources
- Analyzing Blueprint Files
- Creating a Build System
- Current Status
- What’s Next?
What’s Next?

- gradle-build-system
- Build Android libraries
- Build Android system
- Contribute back clang cross-compiler
- Contribute back soong build system and importer