Package Management for Android C++ *by Alex Cohn*

Why Prefab?

Android++ (reminder)

- Android OS is a Linux distribution according to the Linux Foundation
- Toolchain: clang version 8.0.7 (9.0.8 in next beta)
- Standard library: LLVM c++
- Build system: Gradle https://developer.android.com/studio/projects/add-native-code
- Java Native Interface libraries used in Java apps
- JNI binaries distribution: Maven AAR dependencies

  https://developer.android.com/studio/projects/android-library

Challenge++ for Android developers

- Most developers only work in Kotlin or Java, but this may be not enough
- Most developers never leave Android Studio, but this may be not enough
- It’s easy to use a prebuilt JNI library, but this may be not enough
- It’s easy to add your C++ code to Android project, but this may be not enough

The challenge: third-party C++ dependencies!

Challenge++ for library authors

- Need to maintain two build systems: **ndk-build** and **CMake**
- Need to support different build platforms, Windows being more painful
- Need to take care of library dependencies
  - Either bundle all dependencies in a huge source tree (and be locked out of updates)
  - Or link to dependencies and let the app developer suffer

We need an integrated and flexible package management system.

Prefab comes to rescue (1/3)

https://android-developers.googleblog.com/2020/02/native-dependencies-in-android-studio-40.html

root/build.gradle

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:4.0.0-beta04'
    }
}
```

app/build.gradle

```
dependencies {
     implementation 'com.arthenica:mobile-ffmpeg-prefab-min:4.3.1'
}
```

Prefab comes to rescue (2/3)

CMakeLists.txt

```
find_package(ffmpeg REQUIRED CONFIG)

add_library(app SHARED app.cpp)

target_link_libraries(app ffmpeg::libavcodec)
```

Android.mk

```
LOCAL_SHARED_LIBRARIES += libavcodec # libavutil will be available, too
$(call import-module,prefab/ffmpeg)
```

Prefab comes to rescue (3/3)

app.cpp

```
#include "libavcodec/avcodec.h"
```

This prefab package used `make install` *to prepare the public headers for* ***ffmpeg****.*

They are attached to ***libavutil*** *library, but are available if you link any of the ffmpeg libraries (e.g.* ***libavcodec****), because they all depend on* ***libavutil*** *and bring it in for you.*

Prefab features

- Packages with public headers and compiled binaries, and maybe more
- Supports static and shared libraries
- Supports external dependencies (see how **curl** brings **OpenSSL**)
- Provides for matching by ABI (x86, ARM64-v8, etc.), OS version, C++ runtime
- Potentially may be extended to other platforms
- Open source Prefab tool provides build system integration (from AAR to **CMake** or to **ndk-build**) is fully integrated into Android Gradle Plugin v.4.0.

While AGP 4.0 is not released

Add following tweaks to your **gradle.properties** file:

```
# Enables Prefab
android.enablePrefab=true
# Work around https://issuetracker.google.com/149575364
android.enableParallelJsonGen=false
# 4.0.0 canary 9 defaults to Prefab 1.0.0-alpha3, which is not the latest.
android.prefabVersion=1.0.0-alpha5
```

Package List

https://github.com/google/prefab/wiki/Package-List as of Feb 24

```
This is a list of known Prefab packages.

  • com.android.ndk.thirdparty:curl
  • com.android.ndk.thirdparty:jsoncpp
  • com.android.ndk.thirdparty:openssl
  • com.google.oboe:oboe
```

Not very impressive… yet.

You can add your favorites to the wish list of 27.

How to prefab your library

Google built some tools: https://android.googlesource.com/platform/tools/ndkports

Export from vcpkg to Prefab: https://github.com/microsoft/vcpkg/pull/10271

Yours truly has recently prefabbed ffmpeg and openh264. See also a small sample.

Don’t need Maven repo to develop: you can use a local AAR module

```
configurations.maybeCreate("default")
artifacts.add("default", file("../build/publications/ffmpeg.aar"))
```

Thank You *(and some final remarks)*

A Prefab-compatible AAR is easy to build: I've used (for different projects) make, bash, CMake, gradle.

It would be nice (in the future) to be able to export your library as a polyglot bundle for both Android library (for JNI consumers) and Prefab library (for native consumers).

Special thanks to *Dan Albert* for inspiration!