Android’s build system is messier than your distro’s

Denis ‘GNUtoo’ Carikli Ricardo ‘Grim’ Cabrita

Replicant

The Build System

Before **Android 7** (2006-2016)

- Makefile based
- Finds and joins all **Android.mk** files in one
- Limited configuration during build
- Error prone
- Slow and unscalable

The Build System

Android.mk file example

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE     := gzip
LOCAL_SRC_FILES := src/test/generic.c
LOCAL_SHARED_LIBRARIES := libz

ifeq ($(TARGET_ARCH),arm)
  LOCAL_SRC_FILES := src/test/arm.c
else
  ifeq($(TARGET_ARCH),x86)
    LOCAL_SRC_FILES := src/test/x86.c
  endif
endif

include $(BUILD_SHARED_LIBRARY)
```

The Build System

From **Android 7** onwards (2016-Now)

- **Soong Build System**
- **Blueprint files** are replacing the old **Makefiles**
- **Blueprint files** are converted to a **Ninja** manifest
- Remaining **Makefiles** get translated to **Ninja** by **Kati**

The Build System

- **Blueprint files** are simple to write
- **Ninja** is fast
- Addresses performance issues
- Falls short of addressing the lack of modulatiy and flexibility

The Build System

Android.bp file example

```
cc_binary {
    name: "gzip",
    srcs: ["src/test/generic.c"],
    shared_libs: ["libz"],
    arch: {
        arm: {
            srcs: ["src/test/arm.c"],
        },
        x86: {
            srcs: ["src/test/x86.c"],
        },
    },
}
```

The Build System

build.ninja file example

```
 description = Install: /home/grim/lineage16/out/target/product/i9305/data/
     nativetest/ltp/testcases/bin/getcwd04
 command = /bin/bash -c "(rm -f /home/grim/lineage16/out/target/product/i9305/data/
     nativetest/ltp/testcases/bin/getcwd04 ) && (cp /home/grim/lineage16/out/target/
     product>
build /home/grim/lineage16/out/target/product/i9305/data/nativetest/ltp/testcases/
    bin/getcwd04: rule60592 /home/grim/lineage16/out/target/product/i9305/obj/
    NATIVE_TESTS/>
rule rule60593)"
```

Workflow

Getting the source tree

Initialize the repository

```
repo init -u https://git.replicant.us/replicant/manifest.git -b replicant-9
```

Workflow

Getting the source tree

Inside a manifest file

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <remote  name="aosp"
           fetch="https://android.googlesource.com"
           revision="refs/tags/android-9.0.0_r44" />
(...)
  <project path="device/samsung/i9305" name="replicant-9/device_samsung_i9305"
      remote="replicant-9" />
  <project path="external/mesa3d" name="replicant-9/external_mesa3d" remote="
      replicant-9" />
  <project path="frameworks/base" name="replicant-9/frameworks_base" remote="
      replicant-9" />
  <project path="packages/apps/Dialer" name="LineageOS/android_packages_apps_Dialer"
       />
(...)
```

Workflow

Getting the source tree

Download the source tree

```
repo sync
```

Contemplate disk usage

```
du -h --summarize
124GiB
```

Workflow

Configuring the build

Initialize the build environment

```
source build/envsetup.sh
```

```
- lunch:   lunch <product_name>-<build_variant>
- croot:   Changes directory to the top of the tree.
- m:       Makes from the top of the tree.
- mm:      Builds all of the modules in the current directory.
- mmm:     Builds all of the modules in the supplied directories.
- cgrep:   Greps on all local C/C++ files.
- jgrep:   Greps on all local Java files.
- resgrep: Greps on all local res/*.xml files.
- godir:   Go to the directory containing a file.
```

Workflow

Configuring the build

Pick your menu

```
lunch replicant_i9305-userdebug
```

Workflow

Build

Start cooking

```
m bacon
```

Workflow

Upload to device

```
ls out/target/product/i9305
```

```
(...)
boot.img
replicant-ota-eng.user.zip
ramdisk.img
ramdisk-recovery.img
recovery.img
system.img
userdata.img
(...)
```

Workflow

Upload to device

Flash the images directly through download mode

```
heimdall flash --BOOT boot.img --SYSTEM system.img
```

Or install through recovery mode

```
adb sideload replicant-ota-eng.user.zip
```

Workflow

Modify a module

```
cd external/mesa3d
vim src/gallium/drivers/llvmpipe/Android.mk
mma
m snod
heimdall flash --SYSTEM system.img
```

Comparison to other distros

- Recipe maintenance
- Software delivery

Comparison to other distros

Recipe maintenance

GNU/Linux: Most distros use package managers

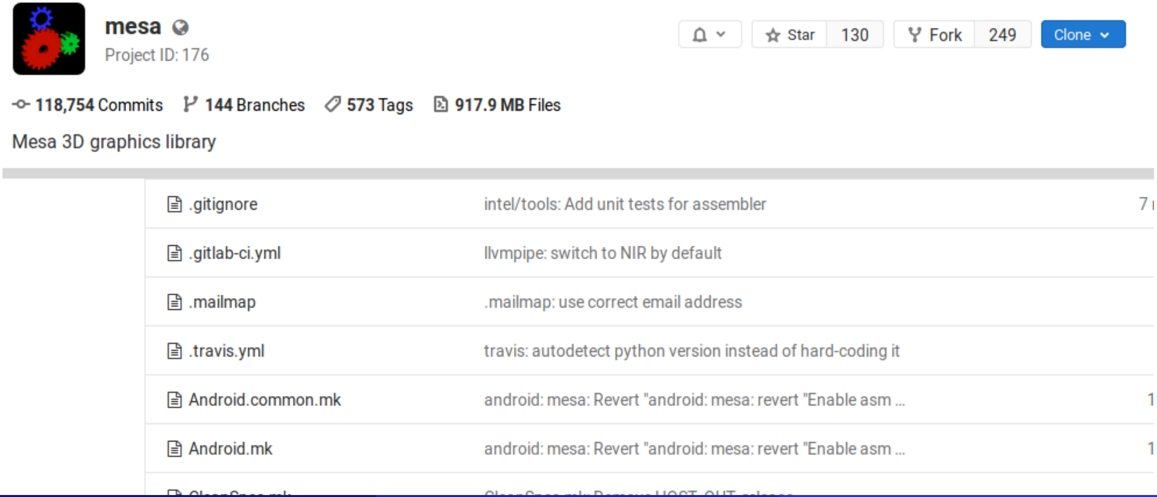
```
pkgbase=mesa
pkgname=(’vulkan-mesa-layer’ ’opencl-mesa’ ’vulkan-intel’ ’vulkan-radeon’ ’libva-
    mesa-driver’ ’mesa-vdpau’ ’mesa’)
pkgdesc="An open-source implementation of the OpenGL specification"
pkgver=19.3.1
license=(’custom’)
(...)

build() {
  arch-meson mesa-$pkgver build \
    -D platforms=x11,wayland,drm,surfaceless \
    -D dri-drivers=i915,i965,r100,r200,nouveau \
(...)
  meson configure build
  ninja -C build
(...)
```

Comparison to other distros

Recipe maintenance

Android: Projects are expected to keep their own **Android.mk**



Project ID: 176

118,754 Commits 144 Branches 573 Tags 917.9 MB Files

Mesa 3D graphics library

.gitignore	intel/tools: Add unit tests for assembler	7
.gitlab-ci.yml	llvmpipe: switch to NIR by default	
.mailmap	.mailmap: use correct email address	
.travis.yml	travis: autodetect python version instead of hard-coding it	
Android.common.mk	android: mesa: Revert "android: mesa: revert "Enable asm ...	1
Android.mk	android: mesa: Revert "android: mesa: revert "Enable asm ...	1
CleanSpec.mk	CleanSpec.mk: Remove HOST_OUT_release	

Comparison to other distros

Recipe maintenance

- Control of the build is out of the hands of the distro
- Troublesome to modify build recipes
- Disregard for the project's prefered build system
- Changes to the recipe have to be made upstream least the distro wishes to maintain yet another fork
- Recipe needs to account for multiple Android versions and distros
- No clear license definition

Comparison to other distros

Software delivery

GNU/Linux: Packages are delivered through software repositories

```
pacman -Ss mesa
extra/mesa 19.3.0-1 [installed: 19.2.7-1]
    An open-source implementation of the OpenGL specification

pacman -S mesa
(...)
Total Download Size:  20.19 MiB
```

Comparison to other distros

Software delivery

Android: Entire system is delivered through a single package

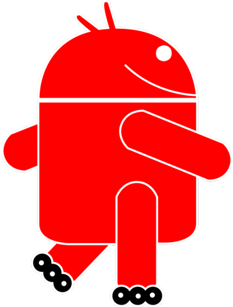
```
du -h replicant-6.0-i9300.zip
269M replicant-6.0-i9300.zip
```

Comparison to other distros

Software delivery

- Only developers can update a single component
- User has to download the entire system for every update
- Updates are rarely delivered in a timely manner
- Best case scenario is the OTA update

Replicant’s stake



Fully free Android distribution running on several devices

Undergoing intense work to make it as **sustainable** as possible

Spending a lot of **development time** on the build system’s quirks

Seeking to discuss on a sustainable way to deal with **external software components**

Replicant's stake

Arguments for supporting **Soong**

- It's fast
- Supports multiple Android versions without convoluted conditional cases
- Works well for Android components

Replicant's stake

Arguments for supporting a **package system**

- More control on the side of the distro
- Less forks to maintain
- More sustainable

Thank you for listening

Come talk to us at the **Replicant assembly** `@` CDC

`https://replicant.us`