Android build system

Nguyen Tran (Nguyen.TranLeHoang@vn.bosch.com)

Oct-2024

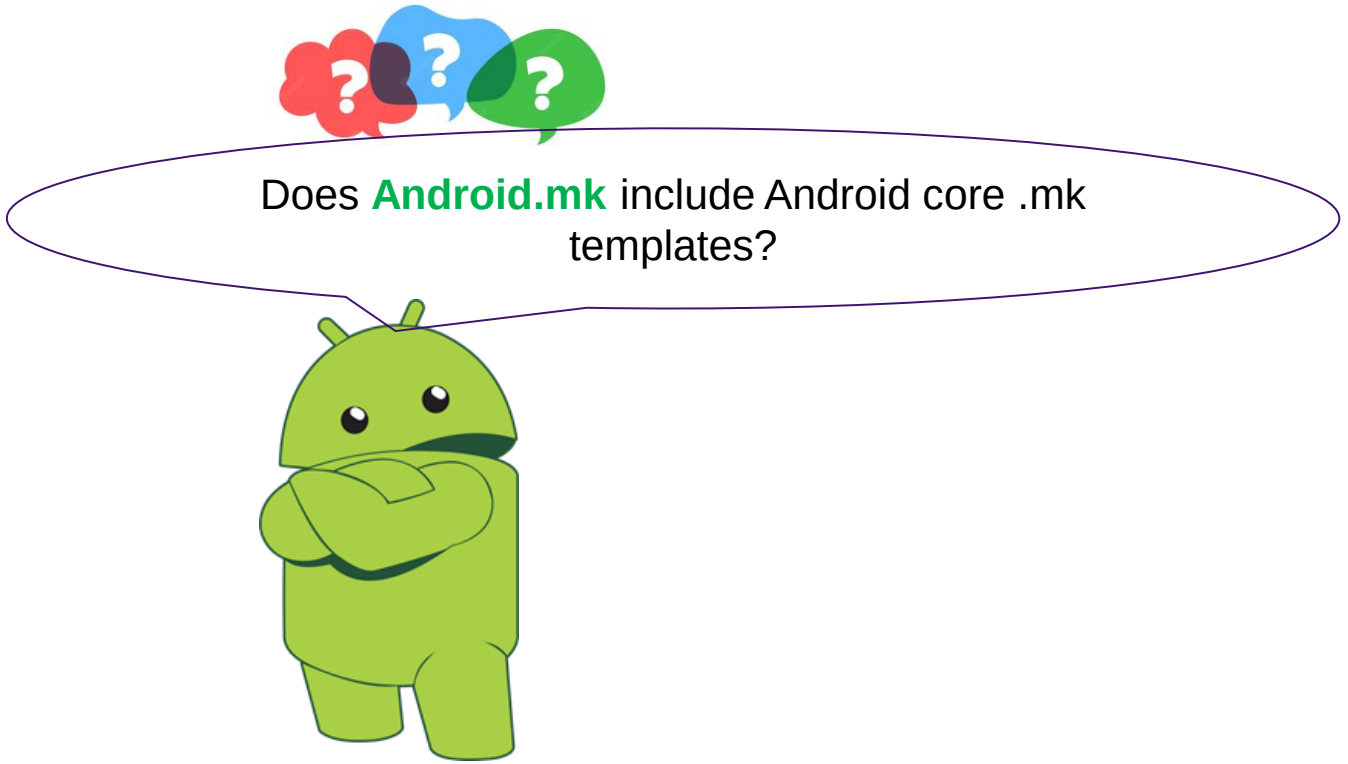
BOSCH

Day 6

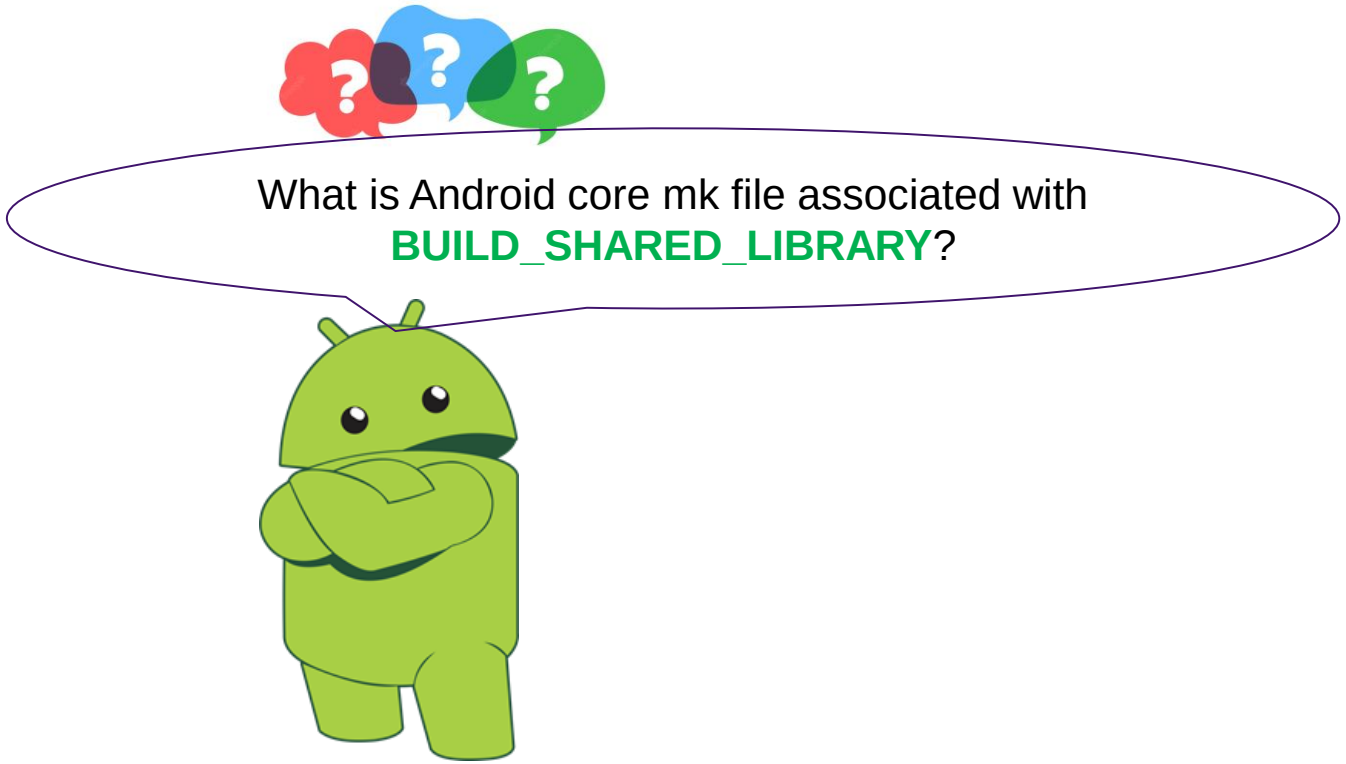
(Android.bp)

QUIZ (1/12)

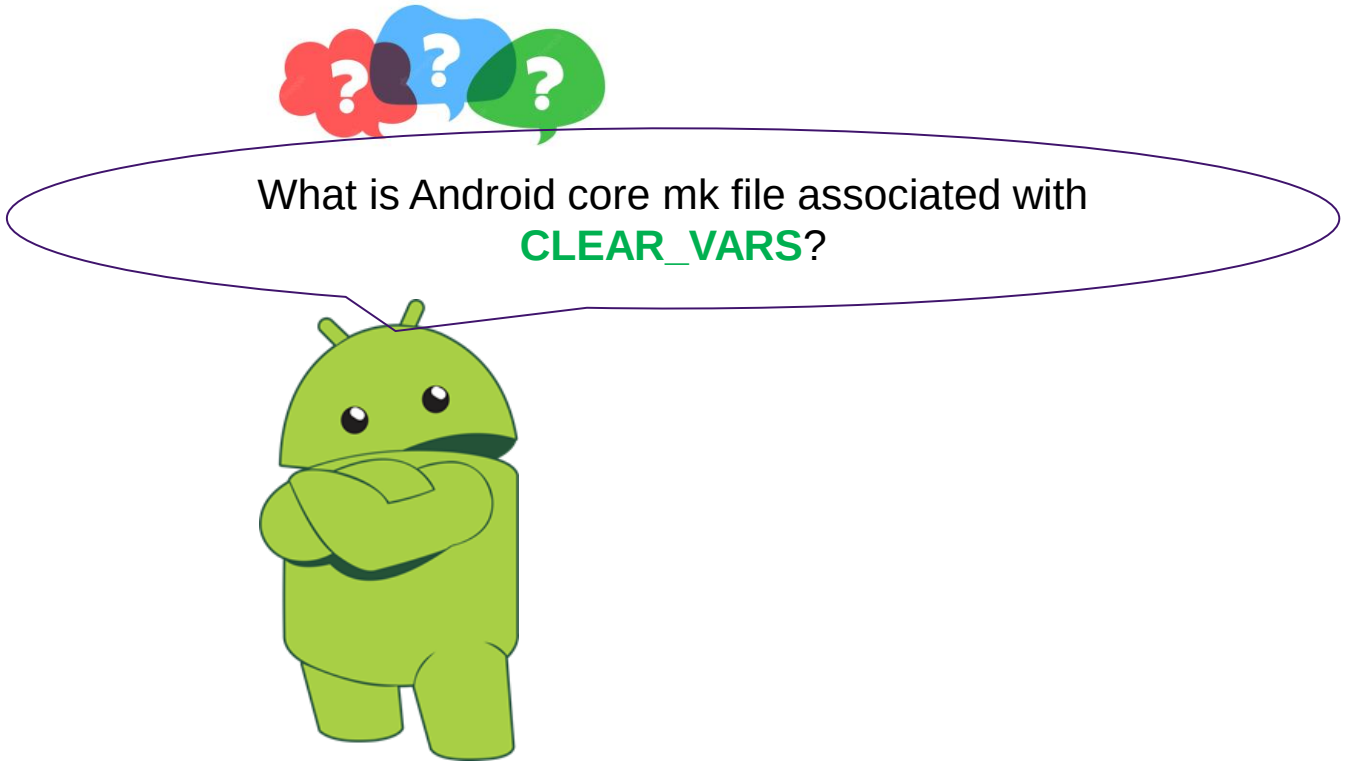
QUIZ (2/12)



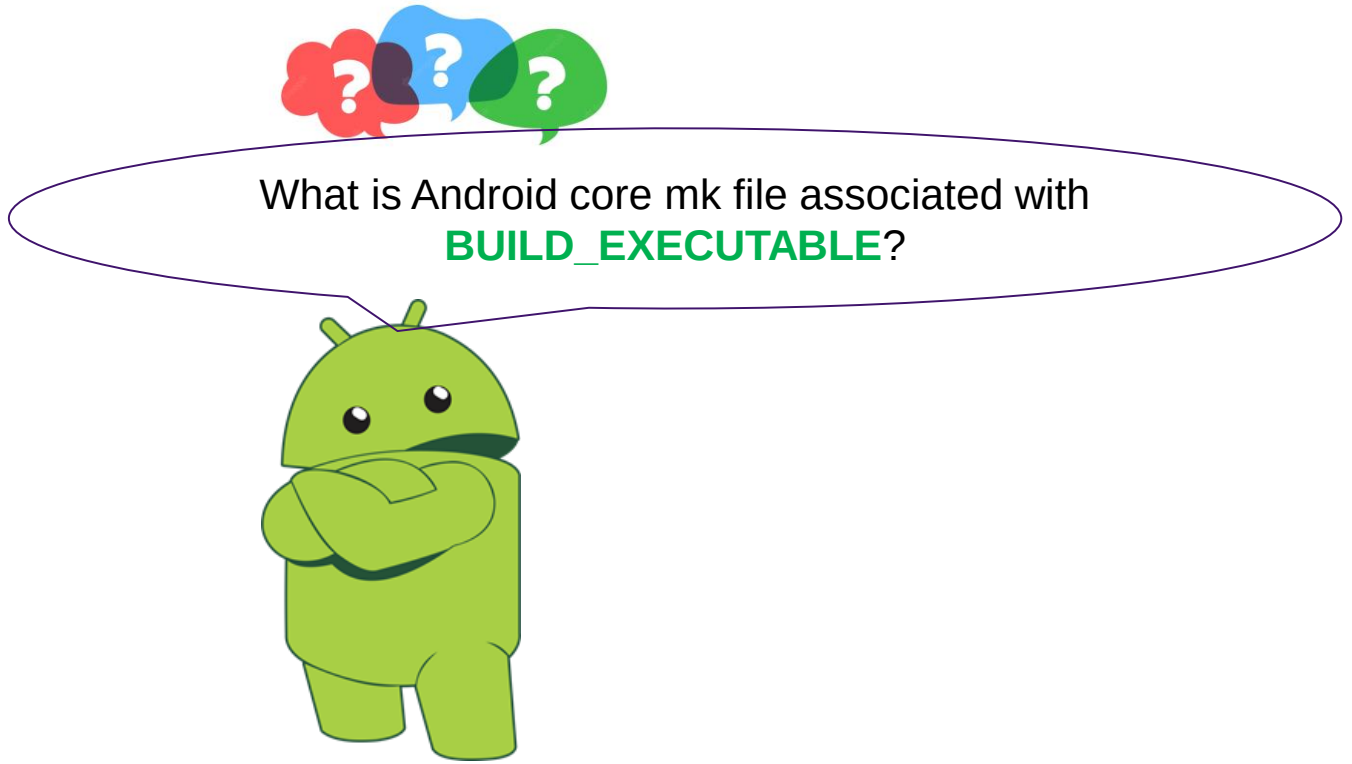
QUIZ (3/12)



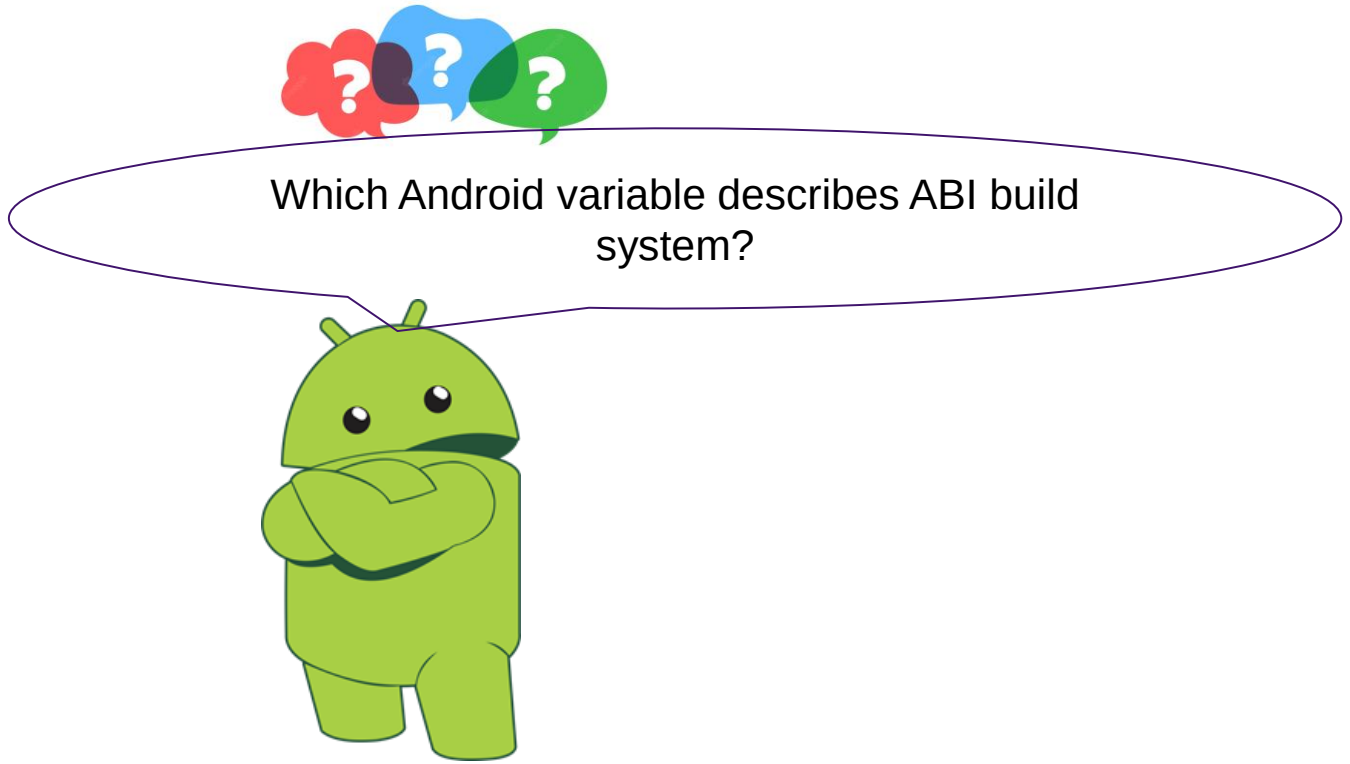
QUIZ (4/12)



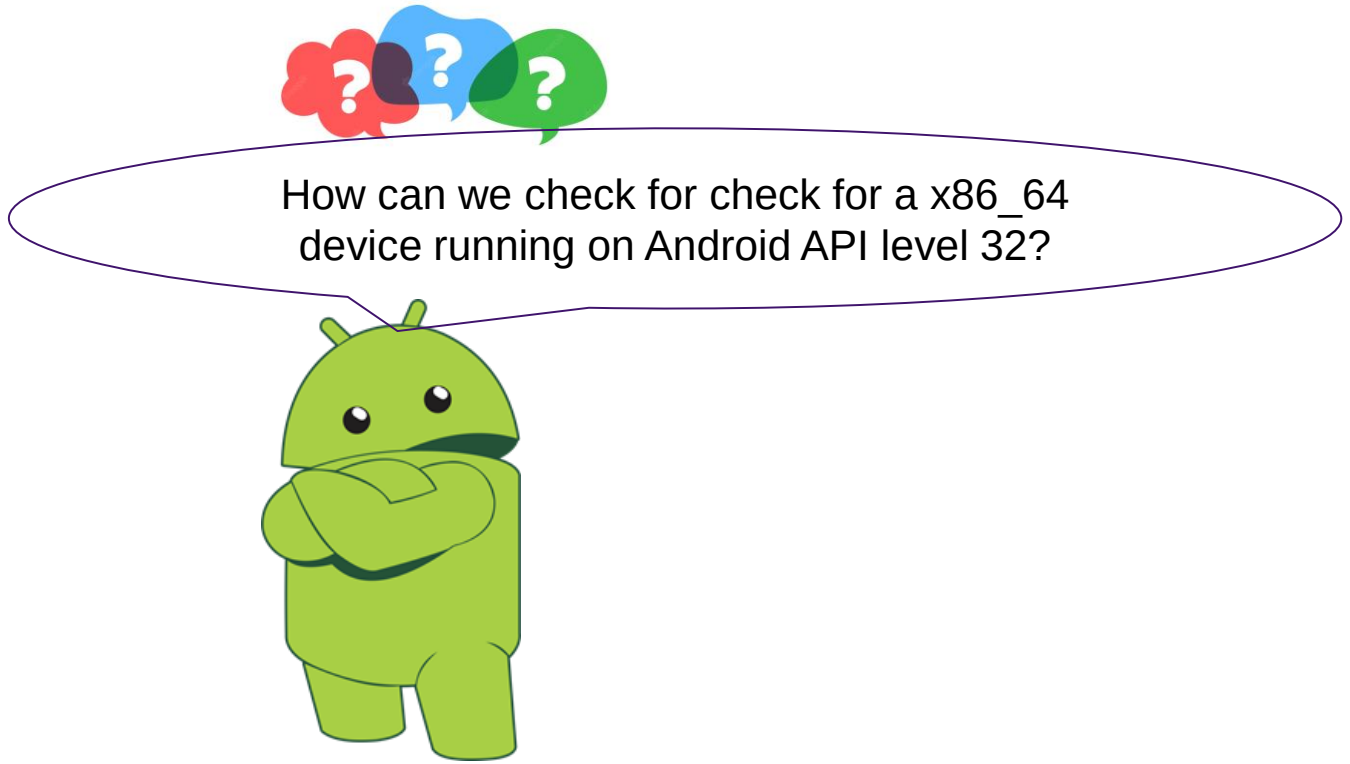
QUIZ (5/12)



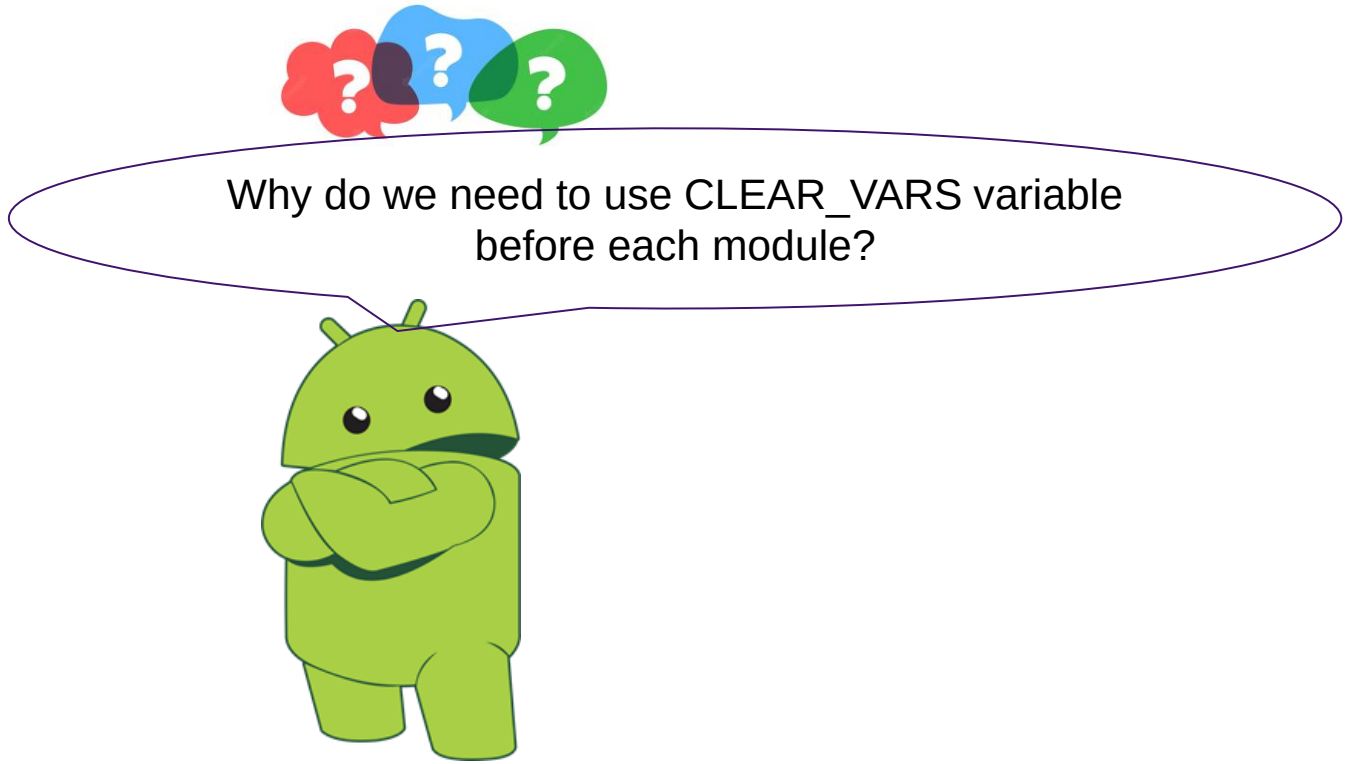
QUIZ (6/12)



QUIZ (7/12)

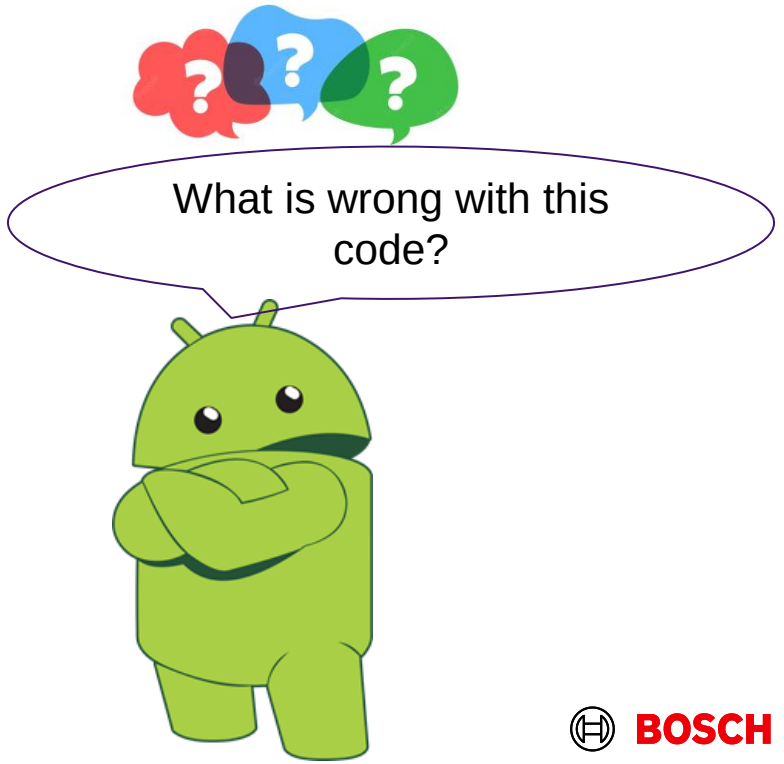


QUIZ (8/12)



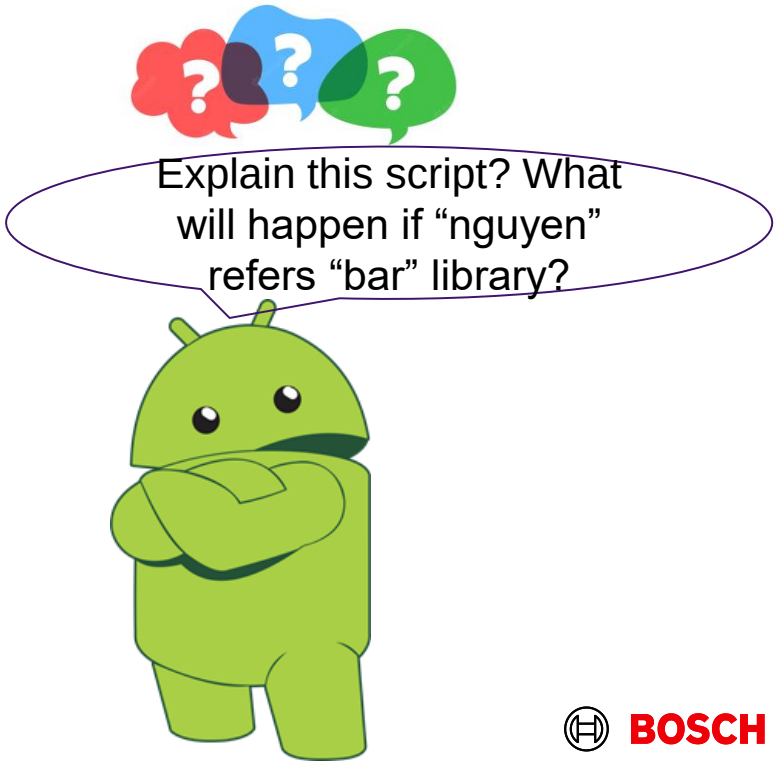
QUIZ (9/12)

```
LOCAL_PATH := $(call my-dir)

# ... declare one module

include $(LOCAL_PATH)/foo/`Android.mk`

LOCAL_PATH := $(call my-dir)

# ... declare another module
```



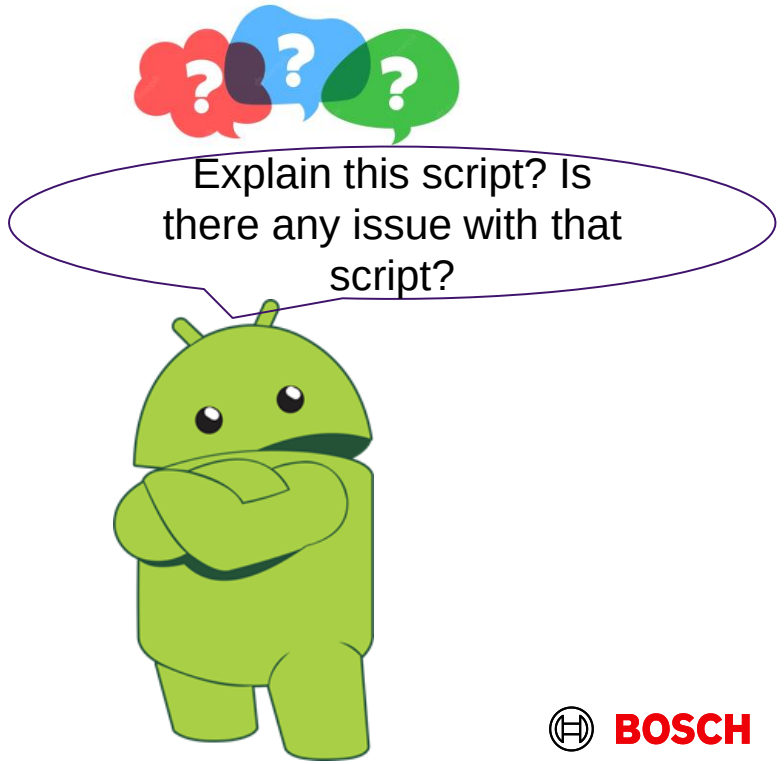
QUIZ (10/12)

```
include $(CLEAR_VARS)
LOCAL_MODULE := foo
LOCAL_SRC_FILES := foo/foo.c
LOCAL_EXPORT_CFLAGS := -DFOO=1
include $(BUILD_STATIC_LIBRARY)


include $(CLEAR_VARS)
LOCAL_MODULE := bar
LOCAL_SRC_FILES := bar.c
LOCAL_CFLAGS := -DBAR=2
LOCAL_STATIC_LIBRARIES := foo
include $(BUILD_SHARED_LIBRARY)
```



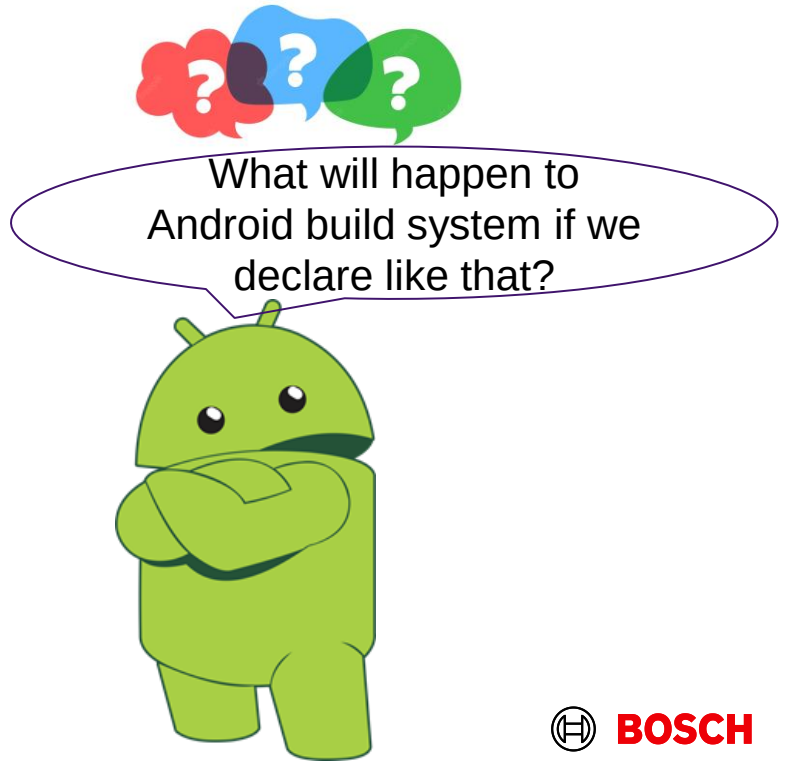
QUIZ (11/12)

```
ifneq ($(filter vsoc_arm64 vsoc_x86 vsoc_x86_64, $(TARGET_BOARD_PLATFORM)),)
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)
include $(LOCAL_PATH)/fetcher.mk

include $(CLEAR_VARS)
include $(LOCAL_PATH)/../host_package.mk

endif

LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)
include $(call all-makefiles-under,$(LOCAL_PATH))
```



QUIZ (12/12)

```
LOCAL_MODULE_TAGS := optional
```



Introduction

- Before the Android 7.0 release, Android used **Android.mk** to describe and execute its build rules.
- **Android.mk** is widely supported and used, but at **Android's scale** became **slow, error prone, unscalable, and difficult to test**.
- **The Soong build system** provides the **flexibility** required for Android builds.
- It replaces **Android.mk** files with **Android.bp** files, which are **JSON-like** simple declarative descriptions of modules to build.

Introduction

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE := libxmlrpc++
LOCAL_MODULE_HOST_OS := linux

LOCAL_RTTI_FLAG := -frtti
LOCAL_CPPFLAGS := -Wall -Werror -fexceptions
LOCAL_EXPORT_C_INCLUDES := $(LOCAL_PATH)/src

LOCAL_SRC_FILES := $(call \
     all-cpp-files-under,src)
include $(BUILD_SHARED_LIBRARY)
```

```
cc_library_shared {
      name: "libxmlrpc++",

      rtti: true,
      cppflags: [
            "-Wall",
            "-Werror",
            "-fexceptions",
      ],
      export_include_dirs: ["src"],
      srcs: ["src/**/*.cpp"],

      target: {
            darwin: {
                  enabled: false,
            },
      },
}
```


BOSCH

File format

- By design, **Android.bp** files are very simple.
- There are **no conditionals or control flow statements** - any **complexity** is handled in build logic written **in Go**.
- It is a **JSON-like** declarative descriptions, which bases name/value pairs.

Modules

- A module in an **Android.bp** file starts with a module type, followed by a set of properties in **name:value,** format
- Every module must have a **name** property, and the **value** must be **unique** across all **Android.bp** files.
- For a list of valid module types and their properties see **$OUT_DIR/soong/docs/soong_build.html**.

```
cc_binary {
    name: "gzip",
    srcs: ["src/test/minigzip.c"],
    shared_libs: ["libz"],
    stl: "none",
}
```

Common module types

- In **Android.bp**, we will build what we need **based on the module type**. The following types are commonly used:
  - **cc_library_shared**: compiled into a **dynamic library**, similar to **BUILD_SHARED_LIBRARY** in **Android.mk**.
  - **cc_binary**: compiled into an **executable file**, similar to **BUILD_EXECUTABLE** in **Android.mk**.
  - **name**: The **name of the compiled module**, similar to **LOCAL_MODULE** in **Android.mk**.
  - **srcs**: **Source files**, similar to **LOCAL_SRC_FILES** in **Android.mk**.

Common module types

- In **Android.bp**, we will build what we need **based on the module type**. The following types are commonly used:
  - **local_include_dirs**: Specifies the **path to find header files**, similar to **LOCAL_C_INCLUDES** in **Android.mk**.
  - **shared_libs**: The **dynamic library that the compilation depends on**, similar to **LOCAL_SHARED_LIBRARIES** in **Android.mk**.
  - **static_libs**: The **static library that the compilation depends on**, similar to **LOCAL_STATIC_LIBRARIES** in **Android.mk**.
  - **cflags**: **Compile Flag**, similar to **LOCAL_CFLAGS** in **Android.mk**.

Common module types

```
cc_library_shared {
    name: "hdmi_cec.yukawa",
    proprietary: true,
    srcs: ["hdmi_cec.c"],
    cflags: ["-Werror"],

    relative_install_path: "hw",

    shared_libs: [
        "liblog",
        "libcutils",
        "libhardware",
    ],

}
```

```
cc_binary {
    name: "hdmicec_test",
    defaults: ["hidl_defaults"],
    vendor: true,
    srcs: ["hdmi_cec_test_hal.c"] + ["hdmi_cec.c"],
    cflags: [
        "-Wno-error",
        "-Wno-unused-parameter",
    ],
    header_libs: ["libhardware_headers"],
    shared_libs: [
        "liblog",
        "libcutils",
    ],

}
```

Common module types

- **android_app**:
  - build the **Android application installation package**.
  - It is a commonly used module type in **Android system application development** and has the same function as **BUILD_PACKAGE** in **Android.mk**.

```
android_app {
    name: "TvProvision",
    srcs: ["**/*.java"],
    product_specific: true,
    sdk_version: "system_current",
    certificate: "platform",
    privileged: true,
    overrides: ["SdkSetup"],
    required: ["privapp_whitelist_com.android.tv.provision"],
    optimize: {
        proguard_flags_files: ["proguard.flags"],
    },
}
```

Common module types

- **java_library**:
  - **build** and **link** the source code into the **device's .jar** file. However, it still has issue:
    - By default, **java_library** has only one variable, which generates a **.jar** package containing **.class** files compiled according to the **device boot classpath**.
    - The resulting jar is **not suitable** for **direct installation** on the device and will usually be used as a **static_libs** dependency of another module.
  - In order to overcome this issue, we should:
    - Specify **installable:true** will generate a .jar file containing the **classes.dex** file, suitable for installation on the device.
    - Specify **host_supported: true** will generate two variables, one compiled according to the **bootclasspath of the device**, and the other compiled according to the **bootclasspath of the host**.

Common module types

- **java_library**:

```
java_library {
    name: "goldfish-fork-sap-api-java-static",
    srcs: ["proto/sap-api.proto"],
    proto: {
        type: "micro",
    },
}
```

Common module types

- **android_library**:
  - **build** the source code along with the android resource files and **link** it into the **device's ".jar" file**.

```
android_library {
    name:"boschcarplaysdk",
    srcs: ["src/**/*.java","transport/**/*.java"],
    manifest: "src/main/AndroidManifest.xml",
    static_libs: ["androidx.appcompat_appcompat","junit","androidx.test.ext.junit","mplay_CinemoSDK"],

    sdk_version: "system_current",
    target_sdk_version: "system_current",
    min_sdk_version: "31",
}
```

Common module types

- **android_app_import**:
  - import a **prebuilt apk** with **additional processing** specified in the module.
  - **DPI-specific apk** source files can be specified using **dpi_variants**.

```
android_app_import {
    name: "example_import",
    apk: "prebuilts/example.apk",
    dpi_variants: {
        mdpi: {
            apk: "prebuilts/example_mdpi.apk",
        },
        xhdpi: {
            apk: "prebuilts/example_xhdpi.apk",
        },
    },
    presigned: true,
}
```

Common module types

- **android_app_certificate**:
  - **android_app_certificate** modules can be referenced by the **certificate's property of android_app modules** to select the signing key.

```
android_app_certificate {
    name: "com.android.bluetooth.updatable.certificate",
    certificate: "com.android.bluetooth.updatable",
}
```

```
apex {
    name: "com.android.bluetooth.updatable",
    enabled: false,

    manifest: "apex_manifest.json",

    native_shared_libs: [
      "libbluetooth_jni",
      "libbluetooth"
    ],
    apps: ["Bluetooth"],

    compile_multilib: "both",

    key: "com.android.bluetooth.updatable.key",
    certificate: ":com.android.bluetooth.updatable.certificate",
}
```

Common module types

- **android_test**:
  - compile test sources and Android resources into an **Android application package** `**.apk**` file and create an `**AndroidTest.xml**` file to allow running the test with `**atest**` or a `**TEST_MAPPING**` file.

```
android_test {
    name: "LocationAccessingApp",
    defaults: ["cts_defaults"],
    srcs: [
        ":location_accessing_app_srcs",
    ],
    aidl: {
        local_include_dirs: ["aidl/"],
    },
    sdk_version: "test_current",
    test_suites: [
        "cts",
        "general-tests",
        "mts",
    ],
}
```

Common module types

- **android_test_helper_app**:
  - compile sources and Android resources into an **Android application package `.apk`** file that will be used by tests but **does not produce an `AndroidTest.xml`** file so the module will **not be run directly as a test**.

```
android_test_helper_app {
    name: "AudioRecorderTestApp_AudioRecord",
    static_libs: ["AudioRecorderTestApp_Base"],
    defaults: ["cts_support_defaults"],
    srcs: ["src/**/*.java"],
    // tag this module as a cts test artifact
    test_suites: [
        "cts",
        "gts",
        "vts10",
        "general-tests",
    ],
    sdk_version: "current",
}
```

Common module types

- **cc_prebuilt_binary**:
  - install a **precompiled executable** in srcs property in the device's directory, for **both the host and device**
  - It is very useful when there's **already a cc_binary with the same name**.
  - This gives developers the flexibility to choose which version to include in their final product.

```
cc_prebuilt_binary {
    name: "linkerconfig",
    prefer: false,
    visibility: ["//visibility:public"],
    apex_available: [
        "//apex_available:platform",
        "com.android.runtime",
    ],
    device_supported: false,
    host_supported: true,
    stl: "c++_static",
    compile_multilib: "64",
    target: {
        host: {
            enabled: false,
        },
        linux_bionic_x86_64: {
            enabled: true,
            srcs: ["linux_bionic/x86_64/bin/linkerconfig"],
        },
        linux_glibc_x86_64: {
            enabled: true,
            srcs: ["linux_glibc/x86_64/bin/linkerconfig"],
        },
    },
}
```

Common module types

- **cc_prebuilt_library**:
  - install a **precompiled shared library** that are listed in the srcs property in the device's directory

```
cc_prebuilt_library {
    name: "art-module-host-exports_libartbase@current",
    sdk_member_name: "libartbase",
    visibility: [
        "//art:__subpackages__",
        "//packages/modules/NetworkStack/tests:__subpackages__",
        "//prebuilts/module_sdk/art/current/host-exports",
        "//prebuilts:__subpackages__",
    ],
    apex_available: [
        "com.android.art",
        "com.android.art.debug",
    ],
    licenses: ["art-module-host-exports_art_license@current"],
    device_supported: false,
    host_supported: true,
    installable: false,
    compile_multilib: "64",
    shared_libs: [
        "libbase",
        "libziparchive",
        "libz",
        "liblog",
        "art-module-host-exports_libartpalette@current",
    ],
    export_include_dirs: [
        "include/art/libartbase",
        "include/system/libbase/include",
        "include/external/fmtlib/include",
    ],
    target: {
        host: {
            enabled: false,
        },
        linux_glibc_x86_64: {
            enabled: true,
            static: {
                srcs: ["x86_64/lib/libartbase.a"],
            },
            shared: {
                srcs: ["x86_64/lib/libartbase.so"],
            },
        },
    },
}
```

Common module types

- **cc_prebuilt_library_shared**:
  - install a **precompiled shared library** that are listed in the srcs property in the device's directory

```
cc_prebuilt_library_shared {
    name: "AdbWinApi",
    defaults: ["AdbWinApi_defaults"],

    export_include_dirs: ["usb/api"],
    srcs: ["prebuilt/usb/AdbWinApi.dll"],
    windows_import_lib: "prebuilt/usb/AdbWinApi.lib",
}
```

Common module types

- **cc_prebuilt_library_static**:
  - install a **precompiled static library** that are listed in the srcs property in the device's directory

```
cc_prebuilt_library_static {
    name: "art-module-sdk_libdexfile_support@current",
    sdk_member_name: "libdexfile_support",
    visibility: ["//visibility:public"],
    apex_available: [
        "//apex_available:platform",
        "com.android.art",
        "com.android.art.debug",
        "com.android.media",
        "com.android.media.swcodec",
        "com.android.runtime",
    ],
    licenses: ["art-module-sdk_art_license@current"],
    host_supported: true,
    installable: false,
    compile_multilib: "both",
    shared_libs: [
        "liblog",
        "libbase",
    ],
    export_include_dirs: [
        "common_os/include/art/libdexfile/external/include",
        "common_os/include/system/libbase/include",
    ],
    target: {
        host: {
            enabled: false,
        },
        android_arm64: {
            srcs: ["android/arm64/lib/libdexfile_support.a"],
        },
        android_x86_64: {
            srcs: ["android/x86_64/lib/libdexfile_support.a"],
        },
        android_arm: {
            srcs: ["android/arm/lib/libdexfile_support.a"],
        },
        android_x86: {
            srcs: ["android/x86/lib/libdexfile_support.a"],
        },
        linux_glibc_x86_64: {
            enabled: true,
            srcs: ["linux_glibc/x86_64/lib/libdexfile_support.a"],
        },
        linux_glibc_x86: {
            enabled: true,
            srcs: ["linux_glibc/x86/lib/libdexfile_support.a"],
        },
    },
}
```

Common module types

- **cc_library_headers**:
  - contain a set of **c/c++ headers** which are imported by other soong cc modules using the **header_libs** property.
  - For best practices, use **export_include_dirs** property or **LOCAL_EXPORT_C_INCLUDE_DIRS** for **Android.mk**.

```
cc_library_headers {
    name: "libinstalld_headers",
    export_include_dirs: ["."],
}
```

Common module types

- **cc_prebuilt_library_headers**:
  - a prebuilt version of **cc_library_headers**

```
cc_prebuilt_library_headers {
    name: "bionic_libc_platform_headers",
    prefer: false,
    visibility: [
        "//art:__subpackages__",
        "//bionic:__subpackages__",
        "//device/generic/goldfish-opengl:__subpackages__",
        "//external/gwp_asan:__subpackages__",
        "//external/perfetto:__subpackages__",
        "//external/scudo:__subpackages__",
        "//frameworks:__subpackages__",
        "//system/core/debuggerd:__subpackages__",
        "//system/core/libcutils:__subpackages__",
        "//system/memory/libmemunreachable:__subpackages__",
        "//system/unwinding/libunwindstack:__subpackages__",
        "//tools/security/sanitizer-status:__subpackages__",
    ],
    apex_available: [
        "//apex_available:anyapex",
        "//apex_available:platform",
        "com.android.media",
        "com.android.runtime",
    ],
    host_supported: true,
    recovery_available: true,
    vendor_available: true,
    product_available: true,
    sdk_version: "current",
    stl: "none",
    system_shared_libs: [],
    export_include_dirs: ["common_os/include/bionic/libc/platform"],
    target: {
        host: {
            enabled: false,
        },
        android: {
            compile_multilib: "both",
        },
        linux_bionic: {
            compile_multilib: "64",
        },
        linux_bionic_x86_64: {
            enabled: true,
        },
    },
}
```

Common module types

- **prebuilt_etc**:
  - **prebuilt artifact** that is installed in **<partition>/etc/<sub_dir>** directory.

```
prebuilt_etc {
    name: "privapp_whitelist_com.android.tv.provision",
    product_specific: true,
    sub_dir: "permissions",
    src: "com.android.tv.provision.xml",
    filename_from_src: true,
}
```

File lists

- Properties can be:
  - Normal **Unix wildcard ***, for example **"*.java".**
  - Even contain **** wildcard** as a path element, which will **match zero or more path elements**. For example, **java/**/*.java** will match:
    - **java/Main.java**
    - and **java/com/android/Main.java**.

```
cc_library_shared {
      name: "libxmlrpc++",

      rtti: true,
      cppflags: [
            "-Wall",
            "-Werror",
            "-fexceptions",
      ],
      export_include_dirs: ["src"],
      srcs: ["src/**/*.cpp"],

      target: {
            darwin: {
                  enabled: false,
            },
      },
}
```

Variables

- An **Android.bp** file may contain top-level variable assignments:

```
gzip_srcs = ["src/test/minigzip.c"],

cc_binary {
    name: "gzip",
    srcs: gzip_srcs,
    shared_libs: ["libz"],
    stl: "none",
}
```

- Variables are **scoped to the remainder of the file** they are declared in, as well as any **child Android.bp files**.
- Variables are **immutable** with one exception - they can be appended to with a **+=** assignment, but **only before** they have been referenced.

Comments

```
package {
    default_applicable_licenses: ["device_generic_goldfish_audio_license"],
}

// Added automatically by a large-scale-change
// See: http://go/android-license-faq
license {
    name: "device_generic_goldfish_audio_license",
    visibility: [":__subpackages__"],
    license_kinds: [
        "SPDX-license-identifier-Apache-2.0",
    ],
    license_text: [
        "NOTICE",
    ],
}
```

```
/*
 * Copyright (C) 2018 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package {
    default_applicable_licenses: ["Android-Apache-2.0"],
}
```


BOSCH

Types

- **Variables and properties** are **strongly typed**
- **Variables dynamically based on the first assignment**, and **properties statically by the module type**.
- The supported types are:
  - Bool (**true** or **false**)

```
proprietary: true,
recovery_available: true,
```

  - Integers (**int**)

```
sdk_version:  28 ,
```

  - Strings (**"string"**)

```
name: "android.hardware.health@2.1-impl-cuttlefish",
```

  - Lists of strings (**["string1", "string2"]**)

```
static_libs: [
    "android.hardware.health@1.0-convert",
    "libbatterymonitor",
    "libhealthloop",
    "libhealth2impl",
],
```

  - Maps (**{key1: "value1", key2: ["value2"]}**)

```
version: {
    py2: {
        enabled: true,
        embedded_launcher: true,
    },
    py3: {
        enabled: false,
    },
},
```

Operators

- **Strings**, **lists of strings**, and **maps** can be appended using the **+ operator**.
- **Integers** can be summed up using the **+ operator**.

```
genrule {
  name: "glesv1_dec_cuttlefish_gensrc",
  srcs: ["GLESv1_dec/*"],
  tools: ["emugen_cuttlefish"],
  cmd: "$(location emugen_cuttlefish) " +
       "-i device/generic/opengl-transport/host/libs/virglrenderer/GLESv1_dec " +
       "-D $(genDir) gles1",
  out: ["gles1_dec.cpp"],
}
```

```
cc_binary {
    name: "hdmicec_test",
    defaults: ["hidl_defaults"],
    vendor: true,
    srcs: ["hdmi_cec_test_hal.c"] + ["hdmi_cec.c"],
    cflags: [
        "-Wno-error",
        "-Wno-unused-parameter",
    ],
    header_libs: ["libhardware_headers"],
    shared_libs: [
        "liblog",
        "libcutils",
    ],

}
```

Conditionals

- Soong **doesn't support conditionals** in **Android.bp** files.
- Instead, **complexity** in build rules that would require conditionals are **handled in Go**, where high-level language features can be used, and implicit dependencies introduced by conditionals can be tracked.
- **Most conditionals** are converted to a **map** property, where one of the values in the map is selected and appended to the top-level properties.

Conditionals

- For example, to support architecture-specific files:

```
cc_prebuilt_library_headers {
    name: "bionic_libc_platform_headers",
    prefer: false,
    visibility: [
        "//art:__subpackages__",
        "//bionic:__subpackages__",
        "//device/generic/goldfish-opengl:__subpackages__",
        "//external/gwp_asan:__subpackages__",
        "//external/perfetto:__subpackages__",
        "//external/scudo:__subpackages__",
        "//frameworks:__subpackages__",
        "//system/core/debuggerd:__subpackages__",
        "//system/core/libcutils:__subpackages__",
        "//system/memory/libmemunreachable:__subpackages__",
        "//system/unwinding/libunwindstack:__subpackages__",
        "//tools/security/sanitizer-status:__subpackages__",
    ],
    apex_available: [
        "//apex_available:anyapex",
        "//apex_available:platform",
        "com.android.media",
        "com.android.runtime",
    ],
    host_supported: true,
    recovery_available: true,
    vendor_available: true,
    product_available: true,
    sdk_version: "current",
    stl: "none",
    system_shared_libs: [],
    export_include_dirs: ["common_os/include/bionic/libc/platform"],
    target: {
        host: {
            enabled: false,
        },
        android: {
            compile_multilib: "both",
        },
        linux_bionic: {
            compile_multilib: "64",
        },
        linux_bionic_x86_64: {
            enabled: true,
        },
    },
}
```

Defaults modules

- A defaults module can be used to repeat the same properties in multiple modules. For example:

```
cc_defaults {
    name: "gzip_defaults",
    shared_libs: ["libz"],
    stl: "none",
}

cc_binary {
    name: "gzip",
    defaults: ["gzip_defaults"],
    srcs: ["src/test/minigzip.c"],
}
```

Packages

- The build is organized into packages where each package is a collection of related files and a specification of the dependencies among them in the form of modules.
- A package is defined as a directory containing a file named **Android.bp**, residing beneath the top-level directory in the build and its name is its path relative to the top-level directory.
- A package includes all files in its directory, plus all subdirectories beneath it, except those which themselves contain an **Android.bp** file.

Packages

- The modules in a package's **Android.bp** and included files are part of the module.

```
.../android/my/app/Android.bp
.../android/my/app/app.cc
.../android/my/app/data/input.txt
.../android/my/app/tests/Android.bp
.../android/my/app/tests/test.cc
```

Referencing Modules

- A module **libfoo** can be referenced by its name

```
cc_binary {
    name: "app",
    shared_libs: ["libfoo"],
}
```

Namespace modules

- Until Android fully converts from **Make** to **Soong**, the **Make** product configuration must specify a **PRODUCT_SOONG_NAMESPACES** value.
- Its value should be a **space-separated** list of namespaces that Soong exports to **Make** to be built by the **m** command.
- Soong provides the ability for modules in different directories to specify **the same name**, as long as each module is declared within a **separate namespace**.

Namespace modules

- A namespace can be declared like this:

```
soong_namespace {
    imports: ["path/to/otherNamespace1", "path/to/otherNamespace2"],
}
```

Exercises

Exercises

1) Write an **Android.bp** file to build C/C++ source code to:
   a) An executable file
   b) A static library
   c) A share libraries
   d) An executable file for API level 32 or above

Thank for your listening!

BOSCH