赶快去学习
发布时间：2020-08-05 02:29

Android P HAL层添加HIDL实例（详细实现步骤）

系统编译 (/search/%E7%B3%BB%E7%BB%9F%E7%BC%96%E8%AF%91/1.htm)

Android P HAL层添加HIDL实例

本文是参照 https://www.jianshu.com/p/b80865c61d8e 教程介绍实现，原理请参考原作者。

本文将介绍如何在P OS上添加HIDL详细实现过程，简单增加seLinux策略使得可以在system_service调用测试，并用模拟器emulator验证。

调用过程为 APP->TestManager->TestService->ITest.hal

文章目录

实现过程

一 、hardware部分

1.1 编写 .hal

.hal的语言格式是C++和Java的结合体。

在 AOSP代码目录 hardware/interfaces/test/1.0/

新建 types.hal (非必要，用于定义结构体，复杂变量可在此定义)

```
//types.hal
package android.hardware.test@1.0;

struct TestID{
        int32_t id;
        string name;
};

struct TestEvent{
        int32_t what;
        string msg;
};
```

新建ITestCallback.hal (非必要，用于回调使用)

```
//ITestCallback.hal
package android.hardware.test@1.0;

interface ITestCallback {
    oneway onTestEvent(TestEvent event);
};
```

新建 ITest.hal (主接口)

```
//ITest.hal
package android.hardware.test@1.0;

interface ITest {
    init(TestID id);
    //无返回值
    helloWorld(string name) generates (string result);
    //变量类型string 不是String
    setCallback(ITestCallback callback) generates (bool res);
    //变量类型bool 不是boolean
    release();
};
```

1.2 使用hidl-gen生成变量

使用hidl-gen的前提是AOSP全编通过，如果之前全编通过可不用再次全编

```
source ./build/envsetup.sh
lunch aosp_car_x86_64-eng
make -j4
make hidl-gen -j4
```

设置临时变量

```
PACKAGE=android.hardware.test@1.0
LOC=hardware/interfaces/test/1.0/default
```

使用hidl-gen生成default目录 里的C++文件

```
hidl-gen -o $LOC -Lc++-impl -randroid.hardware:hardware/interfaces -randroid.hidl:system/libh
idl/transport $PACKAGE
```

使用hidl-gen生成default目录 里的Android.bp文件

```
hidl-gen -o $LOC -Landroidbp-impl -randroid.hardware:hardware/interfaces -randroid.hidl:syste
m/libhidl/transport $PACKAGE
```

使用update-makefiles.sh生成1.0目录下的Android.bp

```
./hardware/interfaces/update-makefiles.sh
```

在default目录创建

此时的目录结构为

```
├── 1.0
│   ├── default
│   │   ├── Android.bp
│   │   ├── Test.cpp
│   │   ├── Test.h
│   │   ├── TestCallback.cpp
│   │   └── TestCallback.h
│   ├── Android.bp
│   ├── ITest.hal
│   ├── ITestCallback.hal
│   └── types.hal
```

TestCallback.cpp和 TestCallback.h没用删掉，并修改default里的Android.bp 删掉TestCallback.cpp

1.3 实现.cpp

Test.h (由hidl-gen工具生成)

line 35 注释打开就是使用passthrough模式

```
#ifndef ANDROID_HARDWARE_TEST_V1_0_TEST_H
#define ANDROID_HARDWARE_TEST_V1_0_TEST_H

#include
#include
#include
#include

namespace android {
namespace hardware {
namespace test {
namespace V1_0 {
namespace implementation {

using ::android::hardware::hidl_array;
using ::android::hardware::hidl_memory;
using ::android::hardware::hidl_string;
using ::android::hardware::hidl_vec;
using ::android::hardware::Return;
using ::android::hardware::Void;
using ::android::sp;

struct Test : public ITest , public Thread{
    // Methods from ::android::hardware::test::V1_0::ITest follow.
    Return init(const ::android::hardware::test::V1_0::TestID& id) override;
    Return helloWorld(const hidl_string& name, helloWorld_cb _hidl_cb) override;
    Return setCallback(const sp<::android::hardware::test::V1_0::ITestCallback>& callback) ov
erride;
    Return release() override;

    // Methods from ::android::hidl::base::V1_0::IBase follow.
    virtual bool threadLoop();
};

// FIXME: most likely delete, this is only for passthrough implementations
// extern "C" ITest* HIDL_FETCH_ITest(const char* name);

}  // namespace implementation
}  // namespace V1_0
}  // namespace test
}  // namespace hardware
}  // namespace android

#endif  // ANDROID_HARDWARE_TEST_V1_0_TEST_H
```

Test.cpp (由hidl-gen工具生成)

简单实现各个方法

```
#define LOG_TAG "Test_cpp"
#include "Test.h"
#include

namespace android {
namespace hardware {
namespace test {
namespace V1_0 {
namespace implementation {

pthread_t pthread;
sp mCallback = nullptr;
std::string mName;
int32_t mID;
bool mExit;

// Methods from ::android::hardware::test::V1_0::ITest follow.
Return Test::init(const ::android::hardware::test::V1_0::TestID& id) {
    mExit = false;
        mName = id.name;
        mID = id.id;
        ALOGD("init:");
        run("test_thread");
    return Void();
}

Return Test::helloWorld(const hidl_string& name, helloWorld_cb _hidl_cb) {
        ALOGD("helloWorld:");
        char buf[100];
        ::memset(buf,0x00,100);
        ::snprintf(buf,100,"Hello World,%s",name.c_str());
        hidl_string result(buf);

        _hidl_cb(result);
    return Void();
}

Return Test::setCallback(const sp<::android::hardware::test::V1_0::ITestCallback>& callback)
 {
        mCallback = callback;
        bool res = false;
    if(mCallback != nullptr) {
        ALOGD("setCallback: done");
                res = true;
    }
    return res;
}

Return Test::release() {
        mExit = true;
        ALOGD("release:");
    return Void();
}

bool Test::threadLoop(){
     static int32_t count = 0;
     TestEvent event;
```

```
     while(!mExit) {
         ::sleep(1);
                 event.msg = mName;
         event.what = count ++;
         if(mCallback != nullptr) {
             mCallback->onTestEvent(event);
         }
     }
     ALOGD("threadLoop: exit");
         return false;
}

// Methods from ::android::hidl::base::V1_0::IBase follow.

//ITest* HIDL_FETCH_ITest(const char* /* name */) {
    //return new Test();
//}
//
}  // namespace implementation
}  // namespace V1_0
}  // namespace test
}  // namespace hardware
}  // namespace android
```

添加启动service

新建android.hardware.test@1.0-service.rc 启动脚本

```
service test_hal_service /vendor/bin/hw/android.hardware.test@1.0-service
    class hal
    user system
    group system
```

新建service.cpp 这里使用绑定式 直通式为注释部分

```
#define LOG_TAG "android.hardware.test@1.0-service"
#include
#include
#include

#include
#include "Test.h"
using android::hardware::configureRpcThreadpool;
using android::hardware::joinRpcThreadpool;
using android::hardware::test::V1_0::implementation::Test;
//using android::hardware::defaultPassthroughServiceImplementation;
//passthrough mode

int main() {
        configureRpcThreadpool(4, true);

    Test test;
    auto status = test.registerAsService();
    CHECK_EQ(status, android::OK) << "Failed to register test HAL implementation";

    joinRpcThreadpool();
    return 0;  // joinRpcThreadpool shouldn't exit
 //   return defaultPassthroughServiceImplementation();
 //passthrough mode
}
```

修改Android.bp

```
cc_binary {
    name: "android.hardware.test@1.0-service",
    relative_install_path: "hw",
    defaults: ["hidl_defaults"],
    proprietary: true,
    init_rc: ["android.hardware.test@1.0-service.rc"],
    srcs: [
                    "Test.cpp",
                "service.cpp",
               ],
    shared_libs: [
                 "libbase",
        "liblog",
        "libdl",
        "libutils",
        "libhardware",
        "libhidlbase",
        "libhidltransport",
        "android.hardware.test@1.0",
    ],
}
```

调用 update-makefiles.sh更新一下

当前目录结构为

```
├── 1.0
│   ├── default
│   │   ├── Android.bp
│   │   ├── Test.cpp
│   │   ├── Test.h
│   │   ├── android.hardware.test@1.0-service.rc
│   │   └── service.cpp
│   ├── Android.bp
│   ├── ITest.hal
│   ├── ITestCallback.hal
│   └── types.hal
```

单编试一下， 出错请检查代码请参考附录解决办法。

```
mmm ./hardware/interfaces/test/1.0
```

```
root@BDJS-PF1LR28T:~/workspace/aosp# mmm ./hardware/interfaces/test/1.0
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=9
TARGET_PRODUCT=aosp_car_x86_64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=x86_64
TARGET_ARCH_VARIANT=x86_64
TARGET_2ND_ARCH=x86
TARGET_2ND_ARCH_VARIANT=x86_64
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-4.4.0-17134-Microsoft-x86_64-Ubuntu-16.04.5-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=PPR1.180610.011
OUT_DIR=out
============================================
ninja: no work to do.
[1/1] out/soong/.bootstrap/bin/soong_build out/soong/build.ninja
No need to regenerate ninja file
out/soong/Android-aosp_car_x86_64.mk was modified, regenerating...

[100% 8/8] Install: out/target/product/generic_x86_64/vendor/bin/hw/android.hardware.test@1.0-service

#### build completed successfully (28:21 (mm:ss)) ####
```


(https://img.it610.com/image/info8/47f593e177ee49e7aa9e8bafc1bcc82c.png)

1.4 VNDK相关

在目录aosp\build\make\target\product\vndk 里

28.txt 和 current.txt 按照字母顺序新增

```
VNDK-core: android.hardware.test@1.0.so
```

二、device部分

由于此次要使用emulator验证，并且lunch的是aosp_car_x86_64-eng

所以在device找到下面目录aosp\device\generic\car\common\manifest.xml

其他device要在对应的目录找到manifest

在manifest.xml添加

```
        <hal  format="hidl">
        <name > android.hardware.test
                                        name>
                                     <transport > hwbinder
                                      transport>
                                         <version > 1.0
                                       version>
                                         <interface >
                                                     name>
                                                                instance>
                                                                   interface>
                <name > ITest              <instance > default              hal>
```

在car.mk添加 启动 test service

```
# Auto modules
PRODUCT_PACKAGES += \
   ……
        android.hardware.test@1.0-service
```

三、SELinux部分——hal service

根据其他博主的文章搜集了一些SELinux的知识点进行补充：
对应sepolicy, Google 也设定了不同的存放目录, 以便进行分离, 以Google 默认的sepolicy 为例.
/system/sepolicy

public: android 和 vendor 共享的sepolicy 定义, 通常情况下, 意味着vendor 开发者可能为此新增一些权限. 一般system/vendor 共用的一些类型和属性的定义, neverallow 限制等会存放于此.

private: 通常意义上的仅限于system image 内部的使用, 不对vendor 开放. 这个只会编译到system image 中.

vendor: 它仅仅能引用public 目录下的相关定义, 这个只会编译到vendor image 中. 但它依旧可以对system image 里面的module 设定sepolicy(对应module 需要在public 下进行声明); 在很大程度上绕过了Google GTS 约束测试.

mapping: 为兼容老版本的sepolicy 而导入, 只有在system image version > vendor version 的时候, 才可能被用到. 即包括两方面, 新版本增加的type , 新版本移除的type, 以及老版本public, 新版本private 等变化的设定, 以兼容老版本.

9.0上Android 安全策略再次加强，hal service需要修改selinux配置

lunch为 aosp_car_x86_64-eng 对应的car device目录下没有BoardConfig.mk，也没有对应的sepolicy

所以只能修改系统的sepolicy

目录为aosp\system\sepolicy

该目录下有vendor public private(注意public 下的修改同样也要修改到prebuilts\api\28.0 下的对应文件，否则编译会报错) 位置找不到参考audiocontrol的位置 或者比较熟悉的hal模块

3.1 vendor 目录

file_contexts 添加

```
/(vendor|system/vendor)/bin/hw/android\.hardware\.test@1\.0-service          u:object_r:hal_test_default_exec:s0
#file_contexts文件保存系统中所有文件的安全上下文定义，每行前半部分是文件的路径，后面是它的安全上下文的定义(hal_test_default_exec)
```

新建 hal_test_default.te
在TE中，所有的东西都被抽象成类型。进程，抽象成类型；资源，抽象成类型。属性，是类型的集合。所以，TE规则中的最小单位就是类型。

#定义一个 名字为 hal_test_default 的type
#TYPE是定义主体和客体所属的类型，对于进程而言，它的类型也称为domian。
#通常主体的type具有domian属性，因此，我们也把主体的type称为domain，将domain设置为hal_test_default的属性，表明zygote是用来描述进程的安全上下文的。

```
# Set a new domain called hal_test_default
type hal_test_default, domain;
# Set your domain as server domain of hal_xxx in which define by AOSP already
hal_server_domain(hal_test_default, hal_test)

# Set your exec file type
type hal_test_default_exec, exec_type, vendor_file_type, file_type;
# Setup for domain transition
init_daemon_domain(hal_test_default)
```

3.2 public 目录

attributes 添加

```
hal_attribute(test);
```

hwservice.te 添加

```
type hal_test_hwservice, hwservice_manager_type;
```

新建 hal_test.te

```
# HwBinder IPC from client to server, and callbacks
binder_call(hal_test_client, hal_test_server)
binder_call(hal_test_server, hal_test_client)

add_hwservice(hal_test_server, hal_test_hwservice)

allow hal_test_client hal_test_hwservice:hwservice_manager find;
```

将以上修改同步到aosp\system\sepolicy\prebuilts\api\28.0\public

3.3 private 目录

hwservice_contexts 添加

```
android.hardware.test::ITest                                    u:object_r:hal_test_hwservic
e:s0
```

private/compat/26.0/26.0.ignore.cil 添加

```
hal_test_hwservice
```

private/compat/27.0/27.0.ignore.cil 添加

```
hal_test_hwservice
```

将以上修改同步到aosp\system\sepolicy\prebuilts\api\28.0\private

四 、客户端实现

4.1 system_service 实现

4.1.1 TestManager端

在目录aosp\frameworks\base\core\java\android\os里 新建test目录

创建ITestService.aidl 对应hal层 四个功能

```
// ITestManager.aidl
package android.os.test;

import android.os.test.ITestEventListener;
// Declare any non-default types here with import statements

interface ITestService {
    /**
     * Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
     void init(int id,String name);
     String helloWorld(String str);
     boolean setTestEventListener(ITestEventListener listener);
     void release();

}
```

创建ITestEventListener.aidl ITestService.aidl 里传递的自定类

```
// ITestEventListener.aidl
package android.os.test;
import android.os.test.TestListenerEvent;
// Declare any non-default types here with import statements

interface ITestEventListener {
    /**
     * Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    void onEvent (inout TestListenerEvent event);
}
```

创建ITestEvent.aidl ITestEventListener.aidl 里传递的 event 类型变量

```
// ITestEvent.aidl
package android.os.test;

// Declare any non-default types here with import statements

parcelable TestListenerEvent;
```

创建TestListerEvent.java

```
package android.os.test;

import android.os.Parcel;
import android.os.Parcelable;

/**
 * android.os.test.TestListenerEvent
 *
 * @author GW00175635
 * @date 2019/7/11
 */
public class TestListenerEvent implements Parcelable {
    private int what;
    private String msg;

    public TestListenerEvent(int what, String msg) {
        this.what = what;
        this.msg = msg;
    }

    public TestListenerEvent(Parcel in) {
        what = in.readInt();
        msg = in.readString();
    }


    public void setWhat(int what) {
        this.what = what;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

    public int getWhat() {

        return what;
    }

    public String getMsg() {
        return msg;
    }

    public static final Creator<TestListenerEvent> CREATOR = new Creator<TestListenerEvent>()
{
        @Override
        public TestListenerEvent createFromParcel(Parcel in) {
            return new TestListenerEvent(in);
        }

        @Override
        public TestListenerEvent[] newArray(int size) {
            return new TestListenerEvent[size];
        }
    };

    @Override
```

```
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(what);
        dest.writeString(msg);
    }

    /**
     *
从parcel中读取，从parcel中读取，顺序与write一致


     * 如果要支持为 out 或者 inout 的定向 tag 的话，需要实现 readFromParcel() 方法
     * @param dest
     */
    public void readFromParcel(Parcel dest) {
        what = dest.readInt();
        msg = dest.readString();
    }

}
```

创建TestManager.java 供上层APP调用的TestManager

```
package android.os.test;

/**
 * android.os.test.TestManager;
 *
 * @author GW00175635
 * @date 2019/7/11
 */

import android.os.RemoteException;
import android.util.Log;

public class TestManager {
    private ITestService mService;
    public static final String TAG = "TestManager";

    public TestManager(ITestService server) {
        Log.d(TAG, "TestManager: ");
        mService = server;
    }

    public void init(int id,String name){
        Log.d(TAG, "init: "+id+" "+name);
        try {
            if (mService != null) {
                mService.init(id,name);
            }
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    public String helloWorld(String str) {
        Log.d(TAG, "helloWorld: "+str);
        try {
            if (mService == null) {
                return null;
            }
            return mService.helloWorld(str);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        return "service connect failed";
    }

    public boolean setTestListener(TestEventListener listener){
        Log.d(TAG, "setTestListener: ");
        try {
            if (mService == null) {
                return false;
            }
            return  mService.setTestEventListener(listener);
        } catch (RemoteException e) {
            e.printStackTrace();
            return false;
        }
    }
```

```
    public void release(){
        Log.d(TAG, "release: ");
        try {
            if(mService != null) {
                mService.release();
            }
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

```
package android.os.test;

/**
 * android.os.test
 *
 *  GW00175635
 *  2019/7/13
 */
public abstract class TestEventListener extends ITestEventListener.Stub  {

}
```

在aosp\frameworks\base\Android.bp里添加

```
                "core/java/android/os/test/ITestEventListener.aidl",
                "core/java/android/os/test/TestListenerEvent.aidl",
                "core/java/android/os/test/ITestService.aidl",
```

4.1.2 TestService端

在目录 aosp\frameworks\base\services\core

Android.bp添加test独有类的引用

```
    static_libs: [
        ……
                "android.hardware.test-V1.0-java",
        ……
    ],
```

在目录 aosp\frameworks\base\services\core\java\com\android\server\test

新建TestService.java

```
package com.android.server.test;


import android.hardware.test.V1_0.ITest;
import android.hardware.test.V1_0.ITestCallback;
import android.hardware.test.V1_0.TestEvent;
import android.hardware.test.V1_0.TestID;
import android.os.RemoteException;
import android.util.Log;

import android.os.test.ITestEventListener;
import android.os.test.ITestService;
import android.os.test.TestListenerEvent;

import java.util.ArrayList;


/**
 * com.android.server.test.TestService
 *
 * @author GW00175635
 * @date 2019/7/11
 */
public class TestService extends ITestService.Stub {
    private String TAG = "TestService";
        private ITest halService ;
    public TestService(){
        try {
            halService = ITest.getService();//获取service
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
    @Override
    public void init(int id, String name) throws RemoteException {
        Log.d(TAG, "init: ");
        TestID testID = new TestID();
        testID.id = id;
        testID.name = name;
        halService.init(testID);
    }

    @Override
    public String helloWorld(String str) throws RemoteException {
        Log.d(TAG, "helloWorld: ");
        return halService.helloWorld(str);
    }

    @Override
    public boolean setTestEventListener(ITestEventListener listener) throws RemoteException {
        Log.d(TAG, "setTestEventListener: ");
        TestCallback testCallback = new TestCallback(listener);
        return halService.setCallback(testCallback);
    }

    @Override
    public void release() throws RemoteException {
```

```
        Log.d(TAG, "release: ");
        halService.release();
    }

    class TestCallback extends ITestCallback.Stub{
        ITestEventListener mITestEventListener;
        TestCallback (ITestEventListener listener){
            mITestEventListener = listener;
        }
        @Override
        public void onTestEvent(TestEvent testEvent) throws RemoteException {
            Log.d(TAG, "onTestEvent: ");
            TestListenerEvent testListenerEvent = new TestListenerEvent(testEvent.what,testEv
ent.msg);
            mITestEventListener.onEvent(testListenerEvent);
        }
    }

}
```

在目录aosp\frameworks\base\core\java\android\content\Context.java里添加

```
        /**
     * {@link android.os.TestManager} for receiving intents at a
     * time of your choosing.
     *
     * @see #getSystemService
     * @see android.os.TestManager
     */
    public static final String TEST_SERVICE = "test";
```

在目录aosp\frameworks\base\core\java\android\app

SystemServiceRegistry.java 里添加

```
import android.os.test.TestManager;
import android.os.test.ITestService;
        ……
                registerService(Context.TEST_SERVICE, TestManager.class,
                new CachedServiceFetcher<TestManager>() {
            @Override
            public HelloManager createService(ContextImpl ctx) {
                IBinder iBinder = ServiceManager.getService(Context.TEST_SERVICE);
                if (iBinder == null) {
                    return null;
                }
                ITestService service = ITestService.Stub
                        .asInterface(iBinder);
                return new TestManager(service);
            }});

            ……
```

在目录aosp\frameworks\base\services\java\com\android\server

SystemServer.java 添加

```
import com.android.server.test.TestService;
    private void startOtherServices(){
                        ……
                        try {
                                Slog.i(TAG, "test Service");
                                ServiceManager.addService(Context.TEST_SERVICE, new TestServi
ce());
                        } catch (Throwable e) {
                                reportWtf("starting TestService", e);
                        }
                        ……
        }
```

4.1.3 添加selinux策略

public 目录

service.te添加

```
type test_service, system_api_service, system_server_service, service_manager_type;
```

将以上修改同步到aosp\system\sepolicy\prebuilts\api\28.0\public

private 目录

service_contexts 添加 和Context 保持一致 TEST_SERVICE = “test”；

```
test                                                                    u:object_r:test_ser
vice:s0
```

system_server.te 添加

```
hal_client_domain(system_server, hal_test)
```

private/compat/26.0/26.0.ignore.cil 添加

```
test_service
```

private/compat/27.0/27.0.ignore.cil 添加

```
test_service
```

将以上修改同步到aosp\system\sepolicy\prebuilts\api\28.0\private

4.2 APP实现调用

4.2.1 APP调用TestManager

在目录aosp\packages\apps\TestAPP下新建

res src 文件夹 src放代码 res放资源 建议在Android studio里编写后复制相关文件到此目录

TestManager可以在Android studio里写个测试用的。

```
package com.gwm.testapp;

import android.app.Activity;

import android.content.Context;
import android.os.Bundle;
import android.os.RemoteException;
import android.os.test.TestManager;
import android.os.test.TestEventListener;
import android.os.test.TestListenerEvent;
import android.view.View;
import android.widget.Button;
import android.util.Log;


/**
 *  GW00175635
 */
public class MainActivity extends Activity {
    TestManager mTestManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mTestManager = (TestManager)getSystemService(Context.TEST_SERVICE);
        Button init = findViewById(R.id.button_init);
        Button hello = findViewById(R.id.button_hello);
        Button set = findViewById(R.id.button_setCallback);
        Button release = findViewById(R.id.button_release);
        init.setOnClickListener(mOnClickListener);
        hello.setOnClickListener(mOnClickListener);
        set.setOnClickListener(mOnClickListener);
        release.setOnClickListener(mOnClickListener);

    }

    View.OnClickListener mOnClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            switch (view.getId()){
                case R.id.button_init:
                    mTestManager.init(123,"name=testAPP");
                    break;
                case R.id.button_hello:
                    String res = mTestManager.helloWorld("HelloFromTestAPP");
                                        Log.d("activity",res);
                    break;
                case R.id.button_setCallback:
                    mTestManager.setTestListener(new TestEventListener(){
                        @Override
                        public void onEvent(TestListenerEvent event) {
                            String msg = event.getMsg();
                            int what = event.getWhat();
                                                Log.d("activity",msg+" "+what);
                        }
                    });
                    break;
```

```
                case R.id.button_release:
                                        mTestManager.release();
                    break;
                default:
                    break;
            }
        }
    };

}
```

4.2.2编写make文件

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

include $(CLEAR_VARS)
LOCAL_CERTIFICATE := platform
LOCAL_MODULE_TAGS := eng
LOCAL_PACKAGE_NAME := TestApp
LOCAL_SRC_FILES := $(call all-subdir-java-files)
LOCAL_PRIVATE_PLATFORM_APIS := true
include $(BUILD_PACKAGE)
```

最后代码目录为

res	2019/7/12 16:00	文件夹	
src	2019/7/12 16:01	文件夹	
Android.mk	2019/7/12 16:02	MK 文件	1 KB
AndroidManifest.xml	2019/7/12 16:00	XML 文件	1 KB

(https://img.it610.com/image/info8/98bba605cc3d4244a851e2df6832b93f.jpg)

五、启动模拟器验证

5.1 替换镜像文件到SDK下

因为改动了系统API，所以在全编之前 执行 make update-api 更新下 API

再执行make 命令

全编之后，在产物目录aosp\out\target\product\generic_x86_64复制以下文件

替换到SDK目录C:\AndroidSDK\system-images\android-28\default\x86_64 里（如果没有该镜像目录，先在AVDManager里下载28 x86_64的镜像）

system-qemu.img 和 vendor-qemu.img 删掉qemu

复制产物目录的data文件夹到SDK目录替换

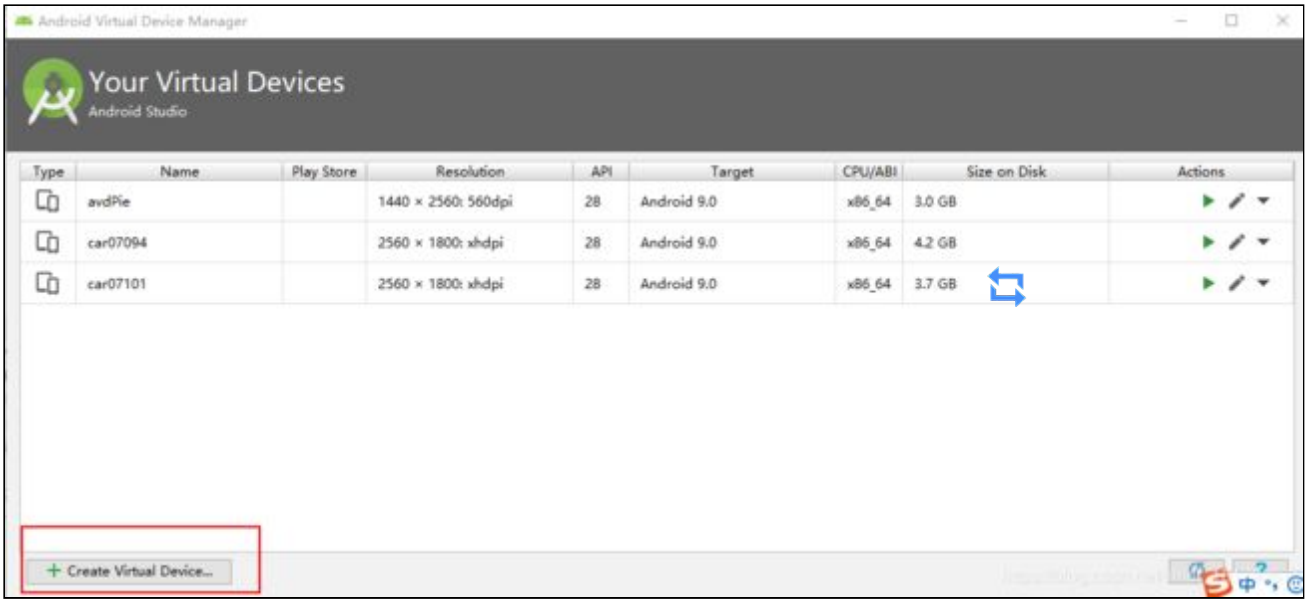
复制产物目录的system\build.prop 到SDK目录替换

在windows平台使用Android studio自带的AVDmanager 启动emulator。

名称	修改日期	类型	大小
advancedFeatures.ini	2019/7/8 14:32	配置设置	1 KB
android-info.txt	2019/7/8 15:40	TXT 文件	1 KB
build_fingerprint.txt	2019/7/12 16:26	TXT 文件	1 KB
build_thumbprint.txt	2019/7/12 16:26	TXT 文件	1 KB
cache.img	2019/7/9 16:12	光盘映像文件	16,384 KB
clean_steps.mk	2019/7/3 15:39	MK 文件	105 KB
config.ini	2019/7/8 14:32	配置设置	1 KB
encryptionkey.img	2019/7/8 14:32	光盘映像文件	1,024 KB
installed-files.json	2019/7/10 9:16	JSON 文件	411 KB
installed-files.txt	2019/7/10 9:16	TXT 文件	132 KB
installed-files-vendor.json	2019/7/10 9:14	JSON 文件	64 KB
installed-files-vendor.txt	2019/7/10 9:14	TXT 文件	20 KB
kernel-ranchu	2019/7/8 14:32	文件	5,839 KB
module-info.json	2019/7/11 17:53	JSON 文件	5,455 KB
previous_build_config.mk	2019/7/3 15:53	MK 文件	1 KB
product_copy_files_ignored.txt	2019/7/12 16:26	TXT 文件	1 KB
ramdisk.img	2019/7/8 15:59	光盘映像文件	1,704 KB
system.img	2019/7/10 9:20	光盘映像文件	2,621,440
system-qemu.img	2019/7/10 9:22	光盘映像文件	2,623,488
userdata.img	2019/7/9 16:12	光盘映像文件	563,200 KB
vbmeta.img	2019/7/8 15:52	光盘映像文件	4 KB
vendor.img	2019/7/10 9:15	光盘映像文件	97,656 KB
vendor-qemu.img	2019/7/10 9:15	光盘映像文件	100,352 KB

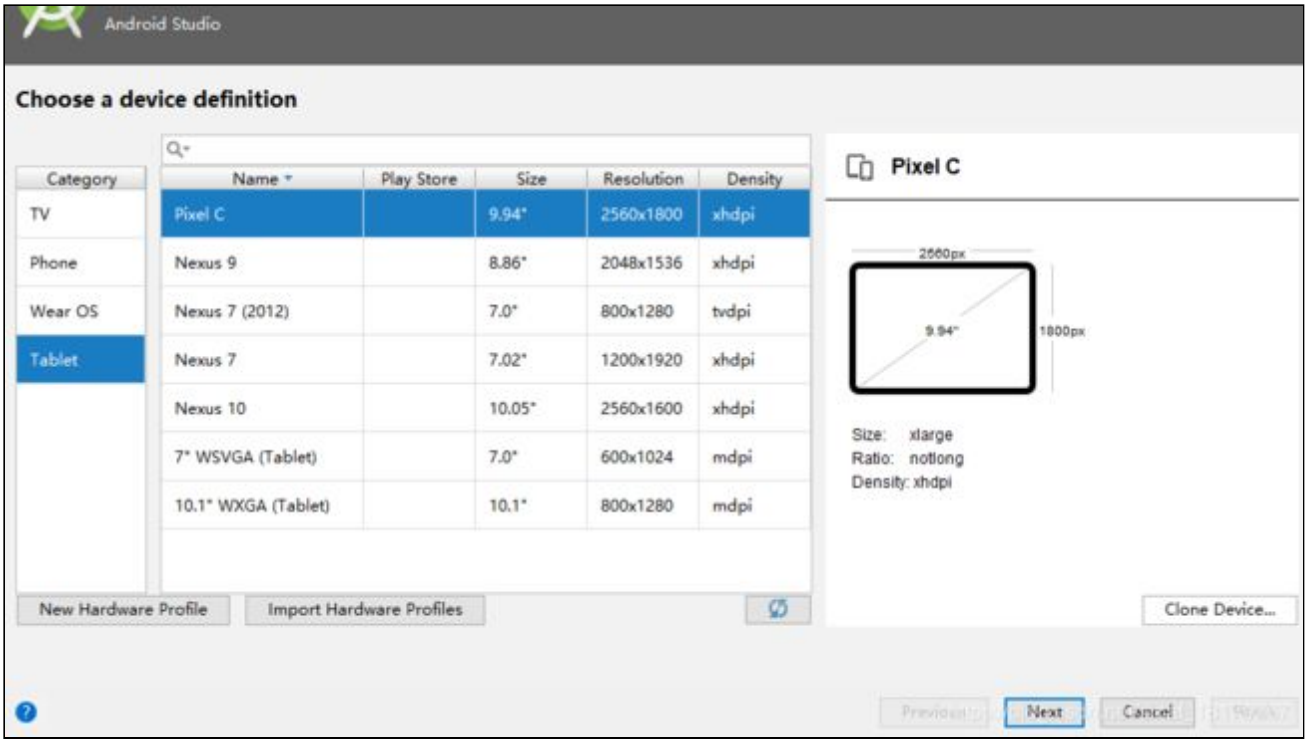
(https://img.it610.com/image/info8/3551a18159ea4061a2a66c0518b1ccd1.jpg)

5.2 启动emulator



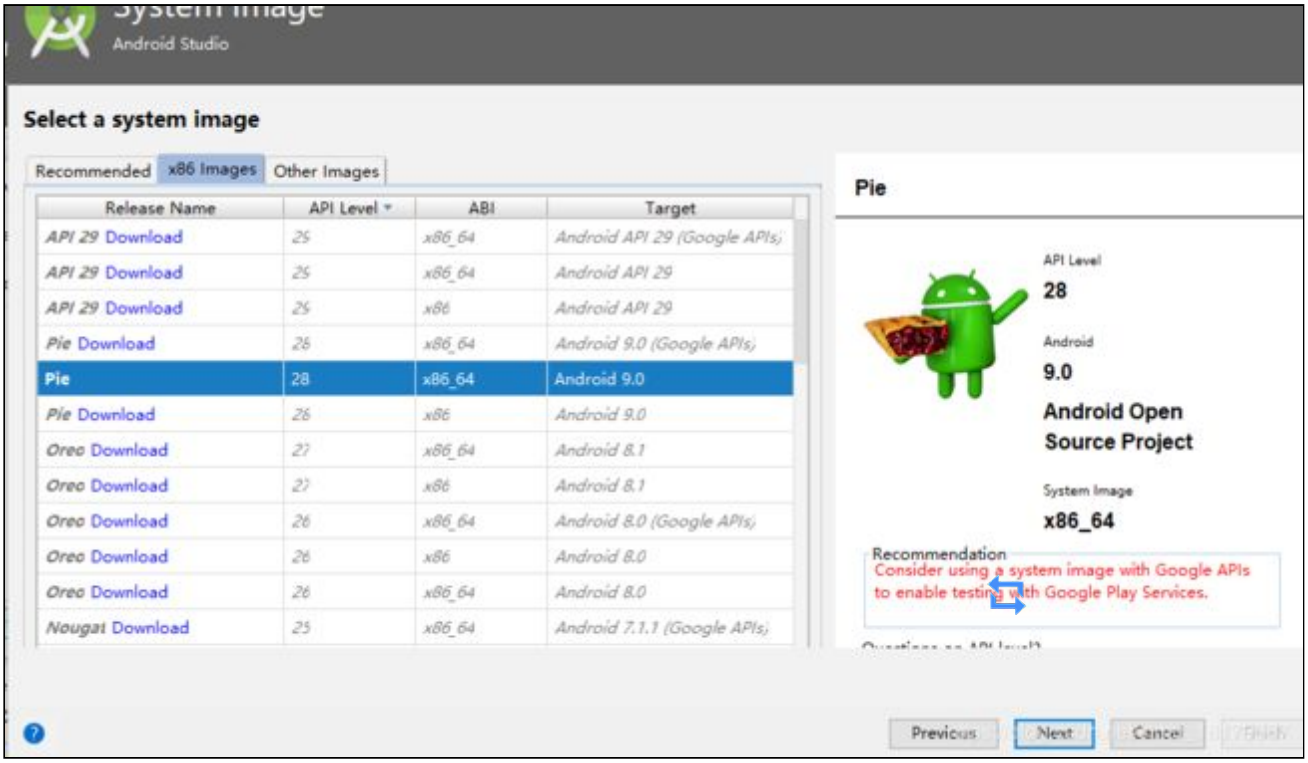
(https://img.it610.com/image/info8/60ec7b1eb78542c883d7221255b8ecb6.jpg)

选个分辨率差不多的



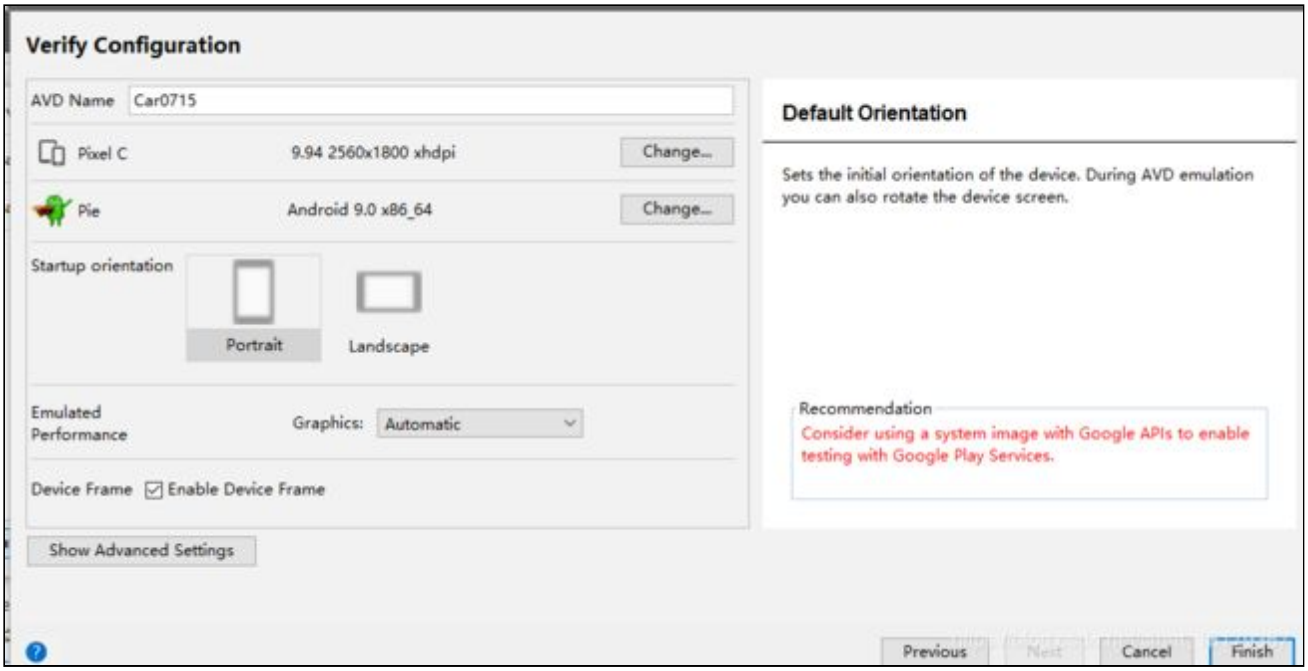
(https://img.it610.com/image/info8/3b898d7b532d45d6b3e8684c5323e850.jpg)

选择刚才替换的镜像



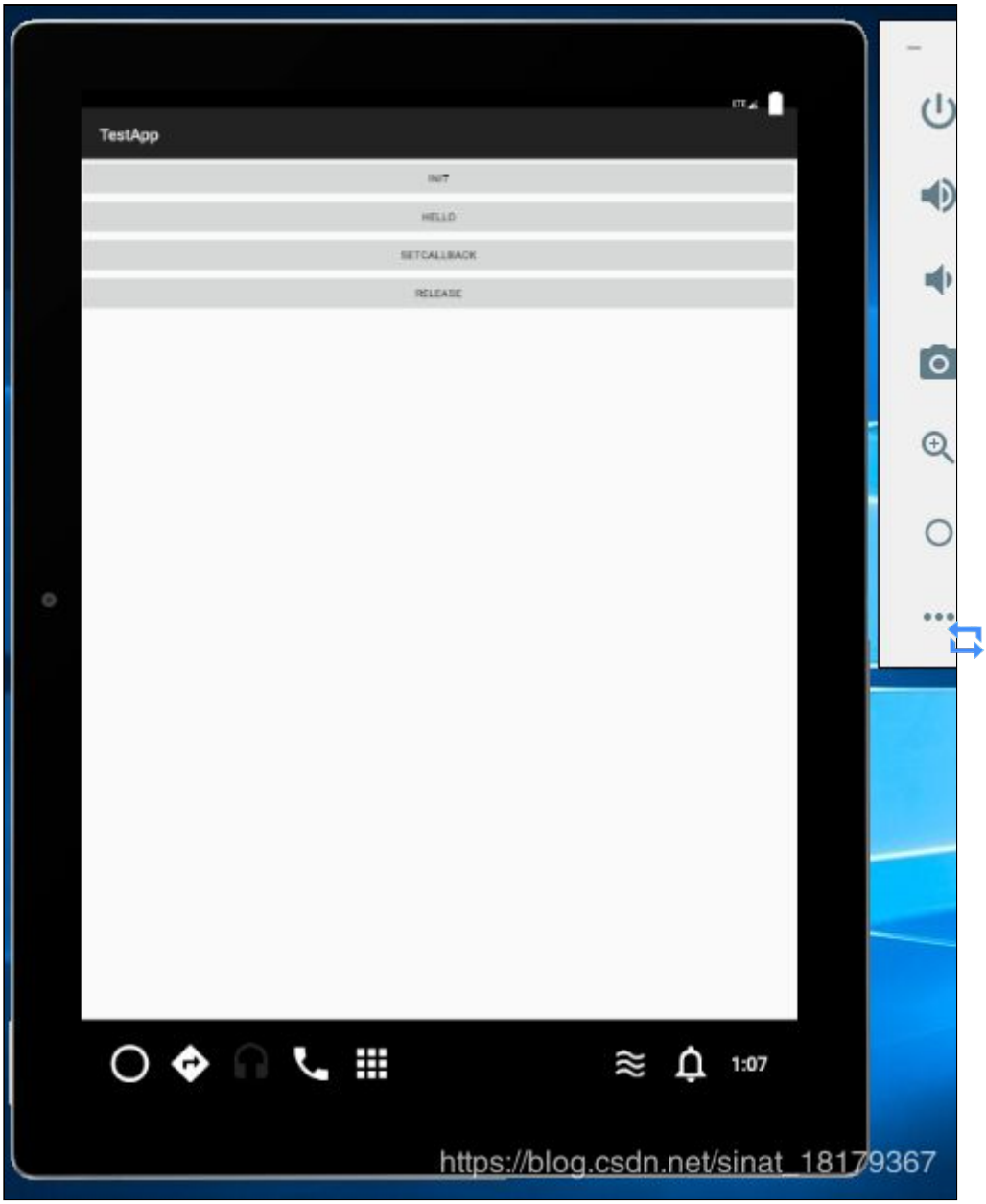
(https://img.it610.com/image/info8/246629e39151466fb261dc515c41264d.jpg)

起一个好记的名字



(https://img.it610.com/image/info8/79cb0bb613084677bb7b91081eeb7b6e.jpg)

使用AVDmanager启动



(https://img.it610.com/image/info8/642ab2cd480c4584921518b0cb639164.jpg)

```
D  4175  4175  TestManager:  setTestListener:
D  2153  3425  TestService:  setTestEventListener:
D  1611  2308  Test_cpp:     setCallback: done
D  4175  4175  TestManager:  init: 123 name=testAPP
D  2153  3425  TestService:  init:
D  1611  2308  Test_cpp:     init:
D  2153  3016  TestService:  onTestEvent: 0 name=testAPP
D  4175  4188  activity:     name=testAPP 0
D  2153  3016  TestService:  onTestEvent: 1 name=testAPP
D  4175  4188  activity:     name=testAPP 1
D  2153  3016  TestService:  onTestEvent: 2 name=testAPP
D  4175  4188  activity:     name=testAPP 2
D  2153  3016  TestService:  onTestEvent: 3 name=testAPP
D  4175  4188  activity:     name=testAPP 3
D  2153  3016  TestService:  onTestEvent: 4 name=testAPP
D  4175  4188  activity:     name=testAPP 4
D  2153  3016  TestService:  onTestEvent: 5 name=testAPP
D  4175  4188  activity:     name=testAPP 5
D  2153  3016  TestService:  onTestEvent: 6 name=testAPP
D  4175  4188  activity:     name=testAPP 6
D  2153  3016  TestService:  onTestEvent: 7 name=testAPP
D  4175  4188  activity:     name=testAPP 7
```

(https://img.it610.com/image/info8/e818ccbd30924442ae79befea68a7d2a.jpg)

5.3 ADB调试

使用CMD命令

在SDK的emulator文件夹目录下

C:\AndroidSDK\emulator>emulator -avd Car0715 -writable-system

```
C:\>cd AndroidSDK\emulator

C:\AndroidSDK\emulator>emulator -avd Car0715 -writable-system
emulator: WARNING: System image is writable
HAX is working and emulator runs in fast virt mode.
```

(https://img.it610.com/image/info8/d79f36dfceb043d2ad873f3505f13eaf.png)

启动avd并让其system可写。

另起一个cmd命令框，就可以有写权限的使用ADB命令

```
C:\Users\GW00175635>adb devices
List of devices attached
emulator-5554   device

C:\Users\GW00175635>adb root

C:\Users\GW00175635>adb remount
remount succeeded

C:\Users\GW00175635>adb install F:\ubuntu_win\aosp\out\target\product\generic_x86_64\system\app\TestApp\TestApp.apk
Success
```


(https://img.it610.com/image/info8/288e2aa0506441e680f2334f8d76f3da.jpg)

附录——错误处理

1.语法错误，结构体少加分号；

```
root@BDJS-PF1LR28T:~/workspace/aosp# hidl-gen -o $LOC -Lc++-impl -randroid.hardware:hardware/
interfaces -randroid.hidl:system/libhidl/transport $PACKAGE
ERROR: missing ; at /mnt/f/ubuntu_win/aosp/hardware/interfaces/test/1.0/types.hal:7.2-1
ERROR: missing ; at /mnt/f/ubuntu_win/aosp/hardware/interfaces/test/1.0/types.hal:12.2-1
hidl-gen F 07-10 11:01:06 30277 30277 Coordinator.cpp:213] Check failed: ret == nullptr
Aborted (core dumped)
```

2.语法错误，忘记import ITestCallbak

```
root@BDJS-PF1LR28T:~/workspace/aosp# hidl-gen -o $LOC -Lc++-impl -randroid.hardware:hardware/
interfaces -randroid.hidl:system/libhidl/transport $PACKAGE
ERROR: Failed to lookup type 'ITestCallback' at /mnt/f/ubuntu_win/aosp/hardware/interfaces/te
st/1.0/ITest.hal:9.31-38
ERROR: Could not parse android.hardware.test@1.0::ITest. Aborting.
```

3.语法错误，boolean改成bool

```
root@BDJS-PF1LR28T:~/workspace/aosp# hidl-gen -o $LOC -Lc++-impl -randroid.hardware:hardware/
interfaces -randroid.hidl:system/libhidl/transport $PACKAGE
ERROR: boolean is a Java keyword and is therefore not a valid identifier at /mnt/f/ubuntu_wi
n/aosp/hardware/interfaces/test/1.0/ITest.hal:9.52-58
ERROR: Could not parse android.hardware.test@1.0::ITest. Aborting.
```

4.玄学bug 怀疑是没有make clean 导致的 也可电脑配置太低 如果之前能全编通过 你也没有修改报错的目录，再次编译就好，然后可能就会发现这个错误没有了，会有新的错误，再次编译直到报错到你的目录或者编译通过，要每次都检查下TARGET_PRODUCT=aosp_car_x86_64 是不是之前你lunch的那个，有时候这个也会变。

```
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=9
TARGET_PRODUCT=aosp_car_x86_64
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=x86_64
TARGET_ARCH_VARIANT=x86_64
TARGET_2ND_ARCH=x86
TARGET_2ND_ARCH_VARIANT=x86_64
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-4.4.0-17134-Microsoft-x86_64-Ubuntu-16.04.5-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=PPR1.180610.011
OUT_DIR=out


[1/1] out/soong/.bootstrap/bin/soong_build out/soong/build.ninja
FAILED: out/soong/build.ninja
out/soong/.bootstrap/bin/soong_build -t -l out/.module_paths/Android.bp.list -b out/soong -n
out -d out/soong/build.ninja.d -o out/soong/build.ninja Android.bp
error: external/llvm/tools/llvm-readobj/Android.bp:5:1: module "llvm-readobj" variant "linux_
glibc_x86_64": module source path external/llvm/tools/llvm-readobj does not exist
ninja: build stopped: subcommand failed.
16:13:52 soong bootstrap failed with: exit status 1


ninja: no work to do.
[1/1] out/soong/.bootstrap/bin/soong_build out/soong/build.ninja
FAILED: out/soong/build.ninja
out/soong/.bootstrap/bin/soong_build -t -l out/.module_paths/Android.bp.list -b out/soong -n
out -d out/soong/build.ninja.d -o out/soong/build.ninja Android.bp
error: external/llvm/tools/llvm-readobj/Android.bp:5:1: module "llvm-readobj" variant "linux_
glibc_x86_64": module source path external/llvm/tools/llvm-readobj does not exist
ninja: build stopped: subcommand failed.
16:13:52 soong bootstrap failed with: exit status 1

ninja: no work to do.
[1/1] out/soong/.bootstrap/bin/soong_build out/soong/build.ninja
FAILED: out/soong/build.ninja
out/soong/.bootstrap/bin/soong_build -t -l out/.module_paths/Android.bp.list -b out/soong -n
out -d out/soong/build.ninja.d -o out/soong/build.ninja Android.bp
error: external/ltp/gen.bp:10404:1: module "ltp_getpriority02" variant "android_x86_x86_64_co
re": module source path external/ltp/testcases/kernel/syscalls/getpriority/getpriority02.c do
es not exist
error: external/ltp/gen.bp:1517:1: module "ltp_uname02" variant "android_x86_x86_64_core": mo
dule source path external/ltp does not exist
error: external/ltp/gen.bp:1517:1: module "ltp_uname02" variant "android_x86_x86_64_core": mo
dule source path external/ltp/testcases/kernel/syscalls/uname/uname02.c does not exist
ninja: build stopped: subcommand failed.
16:16:32 soong bootstrap failed with: exit status 1

16:23:33 Could not create module-finder: finder encountered 1 errors: [/mnt/f/ubuntu_win/aos
```

```
p/external/clang/test/Driver/Inputs/mips_cs_tree/lib/gcc/mips-linux-gnu/4.6.3/include-fixed/n
an2008/el: lstat /mnt/f/ubuntu_win/aosp/external/clang/test/Driver/Inputs/mips_cs_tree/lib/gc
c/mips-linux-gnu/4.6.3/include-fixed/nan2008/el: no such file or directory]
```

5.vndk相关

出现下面错误，删除aosp\out\target\product\generic_x86_64\obj\PACKAGING\vndk_intermediates里的libs.txt，保证aosp\build\make\target\product\vndk 里的28.txt current.txt相同 注意要保持字母顺序添加

```
FAILED: out/target/product/generic_x86_64/obj/PACKAGING/vndk_intermediates/check-list-timesta
mp
/bin/bash -c "(( diff --old-line-format=\"Removed %L\"    --new-line-format=\"Added %L\"
--unchanged-line-format=\"\"\"    build/make/target/product/vndk/28.txt out/target/product/gene
ric_x86_64/obj/PACKAGING/vndk_intermediates/libs.txt          || ( echo -e \" error: VNDK lib
rary list has been changed.\\n\" \"       Changing the VNDK library list is not allowed in AP
I locked branches.\"; exit 1 )) ) && (mkdir -p out/target/product/generic_x86_64/obj/PACKAGIN
G/vndk_intermediates/ ) && (touch out/target/product/generic_x86_64/obj/PACKAGING/vndk_interm
ediates/check-list-timestamp )"
Added VNDK-core: android.hardware.test@1.0.so
 error: VNDK library list has been changed.
        Changing the VNDK library list is not allowed in API locked branches.
```

6.API相关 更新下api make update-api

```
[ 21% 321/1522] Checking API:  checkpublicapi-current
FAILED: out/target/common/obj/PACKAGING/checkpublicapi-current-timestamp
/bin/bash -c "(( out/host/linux-x86/bin/apicheck -JXmx1024m -J\"classpath out/host/linux-x86/
framework/doclava.jar:out/host/linux-x86/framework/jsilver.jar:prebuilts/jdk/jdk8/linux-x86/l
ib/tools.jar:)\"  -error 2 -error 3 -error 4 -error 5 -error 6 -error 7 -error 8 -error 9 -er
ror 10 -error 11 -error 12 -error 13 -error 14 -error 15 -error 16 -error 17 -error 18 -error
19 -error 20 -error 21 -error 23 -error 24 -error 25 -error 26 -error 27  frameworks/base/ap
i/current.txt  out/target/common/obj/PACKAGING/public_api.txt  frameworks/base/api/removed.tx
t  out/target/common/obj/PACKAGING/removed.txt || (  cat build/make/core/apicheck_msg_curren
t.txt ; exit 38 ) ) ) && (mkdir -p out/target/common/obj/PACKAGING/ ) && (touch out/target/co
mmon/obj/PACKAGING/checkpublicapi-current-timestamp )"
out/target/common/obj/PACKAGING/public_api.txt:9538: error 5: Added public field android.cont
ent.Context.TEST_SERVICE
out/target/common/obj/PACKAGING/public_api.txt:33637: error 2: Added package android.os.test

******************************
You have tried to change the API from what has been previously approved.

To make these errors go away, you have two choices:
   1) You can add "@hide" javadoc comments to the methods, etc. listed in the
      errors above.

   2) You can update current.txt by executing the following command:
         make update-api

      To submit the revised current.txt to the main Android repository,
      you will need approval.
******************************
```

7 AIDL 编译错误 Android.bp 里不需要添加aidl的自定义变量类 如 parcelable TestListenerEvent;

```
Exception in thread "main" java.nio.file.NoSuchFileException: out/soong/.intermediates/framew
orks/base/framework/android_common/gen/aidl/frameworks/base/core/java/android/os/test/TestLis
tenerEvent.java
        at java.base/sun.nio.fs.UnixException.translateToIOException(UnixException.java:92)
        at java.base/sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:111)
        at java.base/sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:116)
        at java.base/sun.nio.fs.UnixFileSystemProvider.newByteChannel(UnixFileSystemProvider.
java:215)
        at java.base/java.nio.file.Files.newByteChannel(Files.java:369)
        at java.base/java.nio.file.Files.newByteChannel(Files.java:415)
        at java.base/java.nio.file.Files.readAllBytes(Files.java:3207)
        at com.google.turbine.main.Main.parseAll(Main.java:105)
        at com.google.turbine.main.Main.compile(Main.java:69)
        at com.google.turbine.main.Main.compile(Main.java:61)
        at com.google.turbine.main.Main.main(Main.java:56)
```

8 资源文件找不到错误 如果检查后不是自己的新添加的图片，直接再次全编。如果是自己的文件检查下是不是用了V7 V4的外部引用包资源主题背景等。

```
checkdir error:  cannot create out/target/common/obj/JAVA_LIBRARIES/android_system_stubs_curr
ent_intermediates/classes/res
                 No such file or directory
                 unable to process res/drawable-xhdpi-v4/btn_check_on_holo_dark.png.
checkdir error:  cannot create out/target/common/obj/JAVA_LIBRARIES/android_system_stubs_curr
ent_intermediates/classes/res
                 No such file or directory
                 unable to process res/drawable-xhdpi-v4/btn_check_on_holo_light.png.
checkdir error:  cannot create out/target/common/obj/JAVA_LIBRARIES/android_system_stubs_curr
ent_intermediates/classes/res
                 No such file or directory
                 unable to process res/drawable-xhdpi-v4/btn_check_on_pressed.png.
checkdir error:  cannot create out/target/common/obj/JAVA_LIBRARIES/android_system_stubs_curr
ent_intermediates/classes/res
                 No such file or directory
                 unable to process res/drawable-xhdpi-v4/btn_check_on_pressed_holo_dark.png.
checkdir error:  cannot create out/target/common/obj/JAVA_LIBRARIES/android_system_stubs_curr
ent_intermediates/classes/res
                 No such file or directory
                 unable to process res/drawable-xhdpi-v4/btn_check_on_pressed_holo_light.png.
```