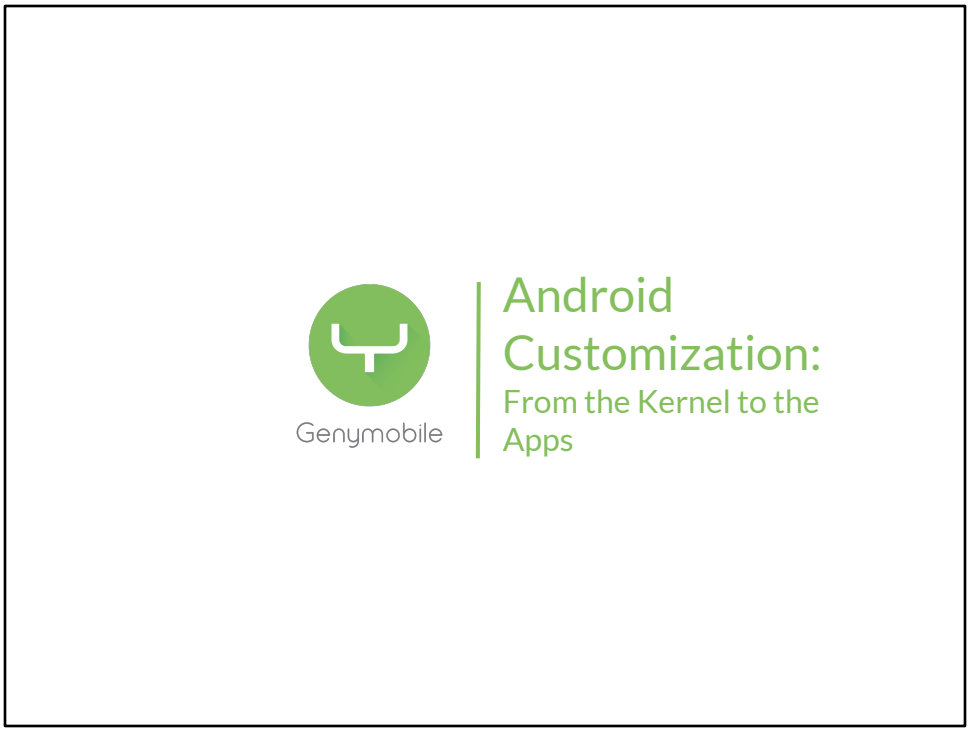


Hi, I am Cédric, I work for Genymobile as a System Engineer
Genymobile is a company specialized in Android. We are based in France (Paris and Lyon) and SF.
We develop and customize android ROM for our customers.
We also have our own products like Genymotion (android emulator, you may have heard of it)

Today I'd like to talk about how to customize a Android system

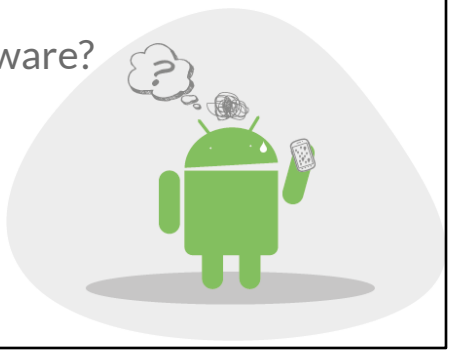
Let’s see what problem we want to solve

Introduction

Genymobile

Android is a “full stack” OS

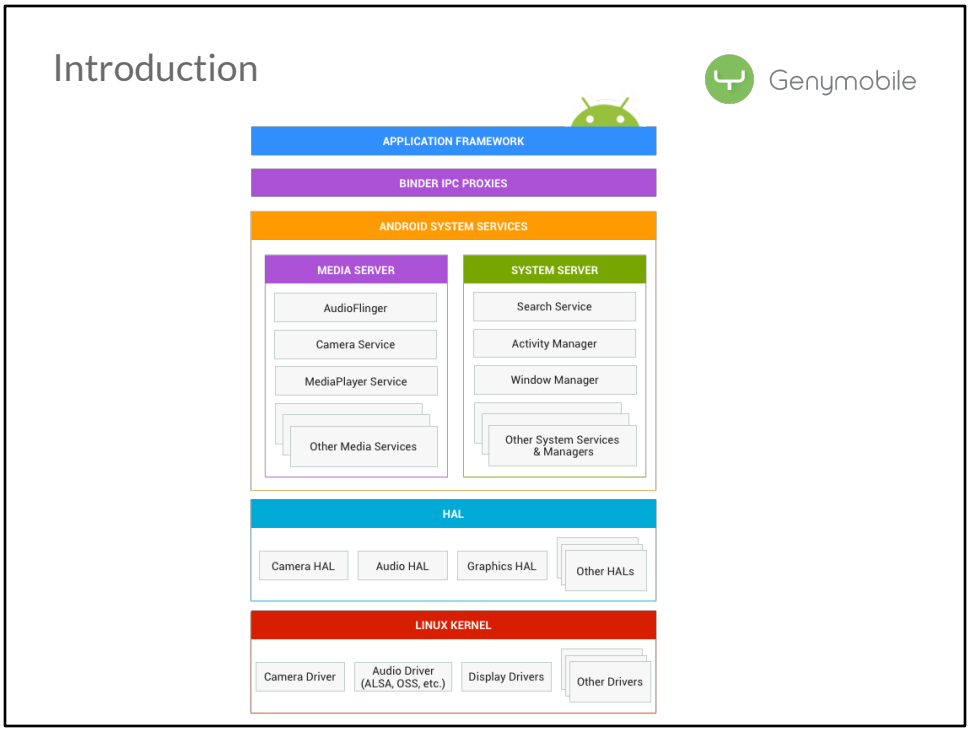
How to use my own hardware?



Android is a full operating system. It come with a SDK to build apps. Every hardware modules can be accessed with a coherent Java API (eg: camera, gps, sensors)
Everything is protected by a Permission mecanism

Very convenient for application developper.

As a linux developper, I’d like to port my own hardware to Android.
Eg: board with a serial port, gpio, ...



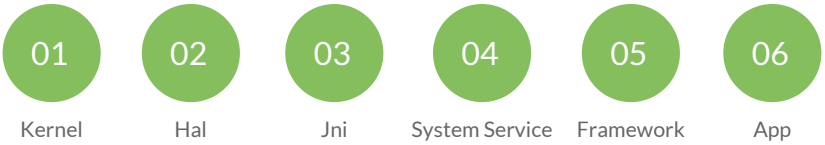
Let’s see how to customize android.
We start from the kernel and go all the way up to the app!

In this presentation we will follow what is done in AOSP.
This mean changing google code.
This is the easiest way, however this can bring problems when we want to port to another android version.

Some other approach need less change in AOSP. To understand how the layers are put together we will keep this method

Summary

Genymobile



Kernel Hal Jni System Service Framework App

01 KERNEL

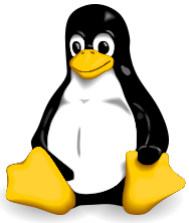
Kernel

Genymobile

Not part of AOSP

Driver: Built-in or Module

GPLv2



Kernel

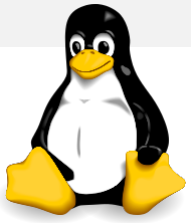
Genymobile

device/<vendor>/<product>/init.<product>.rc

```
on boot
    insmod /system/lib/modules/abs.ko
    chown system system /dev/abs
    chmod 0600 /dev/abs
```

device/<vendor>/<product>/device.mk

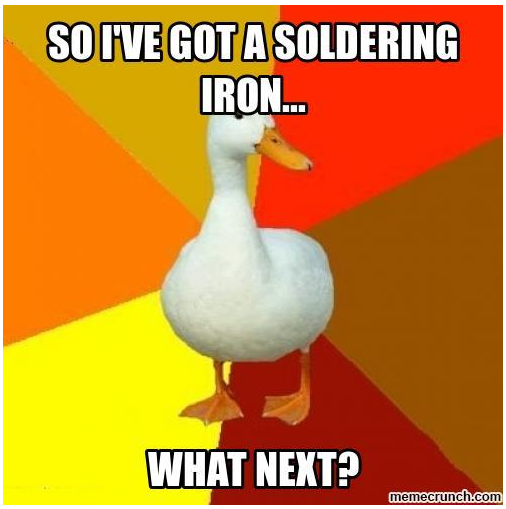
```
PRODUCT_COPY_FILES := \
  device/<vendor>/<product>/abs.ko:system/lib/modules/abs.ko
```



Every file that is used to build and customize you device go to “device”
This module will create a “character device”
We do not want to give access to the device to every application, we restrict to the system user.
We also do not want our app to run as system.

Kernel

Genymobile

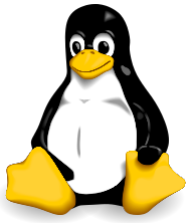


No demo with real hardware

Kernel

Genymobile

```
# ls -l /dev/abs
crw------- system   system   249,   0  abs
# echo “Hello ABS” > /dev/abs
# cat /dev/abs
hELLO abs
```



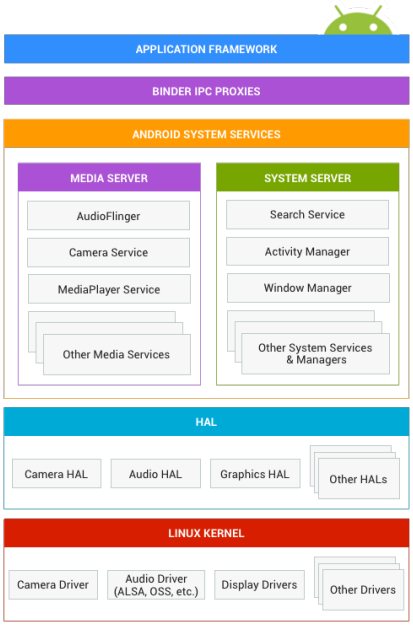
Our “device” is a simple module that convert upper case to lower case

We want to protect access to the device, only “system” is allowed to r/w. Of course our application will not have system permission.

We need the glue to let system_server use it

Kernel

Genymobile



02 HAL: Hardware Abstraction Layer

Hal

Hardware Abstraction Layer

Genymobile

C library

Expose hardware feature

Part of AOSP, often closed source

Hal

Hardware Abstraction Layer

Genymobile

libabs.h

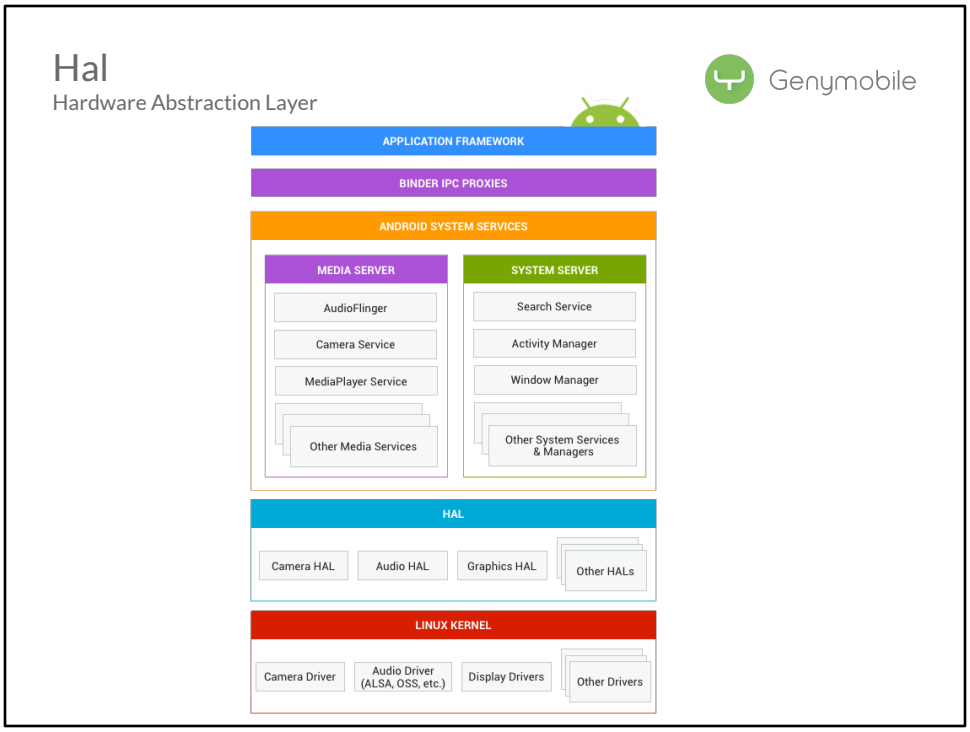
```
ssize_t abs_getdata(void *buf, size_t count);

ssize_t abs_putdata(const void *buf, size_t count);

void abs_clear(void);
```

=> *libabs.so*

With this hardware, I want to get some data (abs_getdata), put new data (abs_putdata) and clear the buffer (abs_clear).
This is here that you will put your device specific code.
This code is not android specific



System server run java code.
We need a bridge between java (system_server) and C (hal) => JNI

03 JNI: Java Native Interface

Jni

Java Native Interface

Genymobile

Simple glue between C and Java

Do not do smart thing here

Jni

Java Native Interface

Genymobile

framework/base/abs/jni/android.abs.Abs.c

```
static void jni_abs_putData(JNIEnv *env, jclass cls, jstring string)
{
        int ret;
        const char *buff = (*env)->GetStringUTFChars(env, string, NULL);
        int length = (*env)->GetStringLength(env, string);
        ret = abs_putdata(buff, length);
        if (ret < 0) {
            ThrowAbsException(env, "fail to put data");
        }
        (*env)->ReleaseStringUTFChars(env, string, buff);
}
```

Expose your hal function through jni
Manage error code with exceptions
Only glue code

Jni

Java Native Interface

Genymobile

framework/base/abs/java/android/abs/Abs.java

```
package android.abs;

public class Abs {
        static {
           System.loadLibrary("abs_jni");
        }

        public native static void clear();
        public native static String getData() throws AbsException;
        public native static void putData(String in) throws AbsException;
}
```

Jni

Java Native Interface

Genymobile

Trick: Add a Main.java

```
package android.abs;
/* @hide */
public class Main {
  public static void main(String[] args) {
    try {
      Abs.putData("Hello ABS");
      String out = Abs.getData();
      System.out.println(out);
      Abs.clear();
    } catch (Exception e) {
      System.out.println(e.toString());
    }
  }
}
```

Jni

Java Native Interface

Genymobile

Trick: Add a Main.java

```
$ dalvikvm -cp /system/framework/framework.jar android.abs.Main
```

Allow you check that everything works from java to the device

Jni

Java Native Interface

Genymobile

04 System Server

System Server

Genymobile

Must Implement AIDL interface

framework/base/abs/java/android/abs/IAbsManager.aidl

```
package android.abs;
/** {@hide} */
interface IAbsManager
{
        void clear();
        String getData();
        void putData(String data);
}
```

System server is called by the application (framework) through the binder protocol
System server run as the “system” user

System Server

Genymobile

framework/base/services/abs/java//android/server/abs/AbsService.java

```
/**  @hide  */
public class AbsService extends IAbsManager.Stub {
  private static final String TAG = "AbsService";
  private Context mContext;

  public AbsService(Context context) {
    mContext = context;
  }

  public String getData() {
    enforceAccessPermission();
    try {
      return Abs.getData();
    } catch(AbsException e) {
      Slog.e(TAG, "cannot getdata");
    }
    return null;
  }
  private void enforceAccessPermission() {
    mContext.enforceCallingOrSelfPermission(android.Manifest.permission.ABS_ACCESS,
                                                        "AbsService");
  }
```

The binder RPC allow us to check the permission of the caller live into service.jar

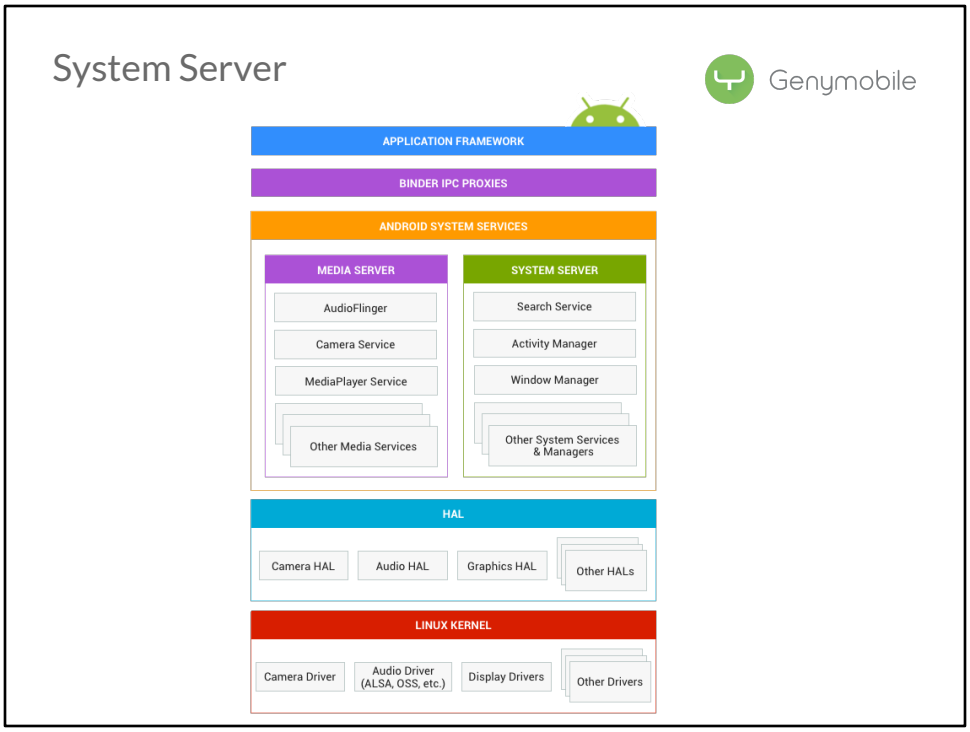
System Server

Genymobile

Hack SystemServer.java

```
private void startOtherServices() {
    …
    try {
        Slog.i(TAG, "Abs Service");
        absService = new AbsService(context);
        ServiceManager.addService(Context.ABS_SERVICE, absService);
    } catch (Throwable e) {
        reportWtf("starting abs Service", e);
    }
}
```

start our service



Note that we decide that our device will be manageable by system_server. However we could have created a special daemon to deal with the device and let system_server talk to the daemon. The daemon could run as root
This is another choice of architecture but do not change much

05 Framework

Framework

Genymobile

frameworks/base/abs/java/android/abs/AbsManager.java

```
package android.abs;

public class AbsManager {
        IAbsManager mService;
        /** @hide   */
        public AbsManager(IAbsManager service) {
             mService = service;
        }

        public String getData() {
          try {
            return mService.getData();
          } catch (RemoteException e) {
            ...
          }
          return null;
        }
}
```

Framework

Genymobile

Hack ContextImpl.java

```
static {
    ...
    registerService(ABS_SERVICE, new ServiceFetcher() {
        public Object createService(ContextImpl ctx) {
            IBinder b = ServiceManager.getService(ABS_SERVICE);
            IAbsManager service = IAbsManager.Stub.asInterface(b);
            return new AbsManager(service);
        }});
}
```

Framework

Genymobile

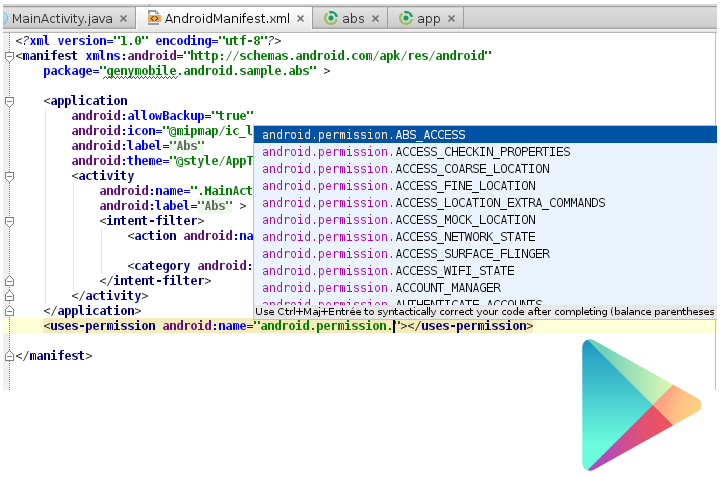
make update-api && make sdk

Configure IDE

06 App

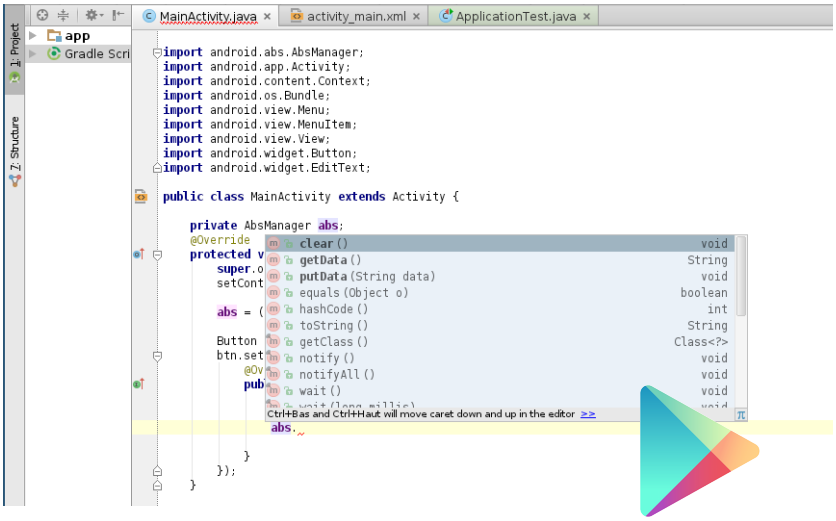
App

Genymobile

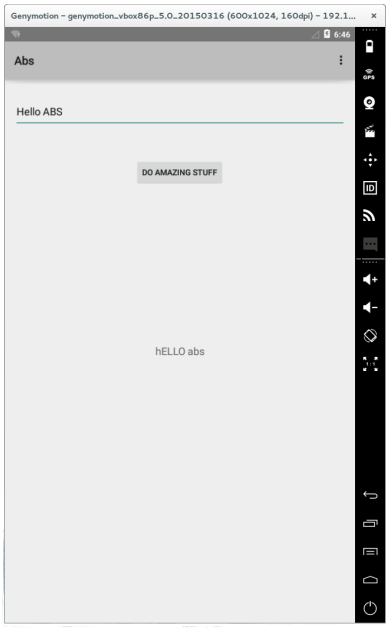


App

Genymobile



App

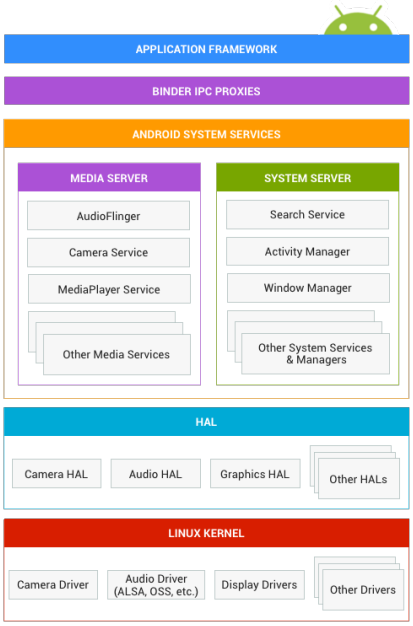


Genymobile

06 Conclusion

Conclusion

Genymobile



Conclusion

Genymobile

Easy to add new driver and expose an “Android API”

Most of the kernel and HAL is reusable

Lot of Glue Code

Conclusion

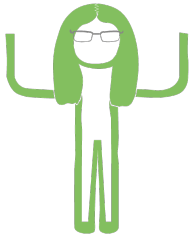
Genymobile

https://thenewcircle.com/s/post/1044/remixing_android

http://processors.wiki.ti.com/index.php/Android-Adding_SystemService

https://source.android.com/devices/

Genymobile



Thank you !

Cédric Cabessa
ccabessa@genymobile.com
www.genymobile.com

https://github.com/CedricCabessa/abs2015

http://goo.gl/nDVszZ