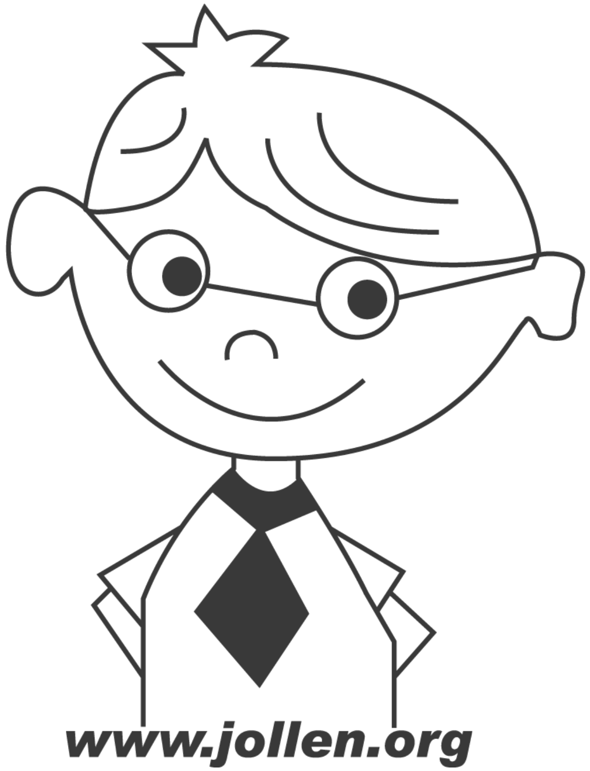
Android 驅動開發關鍵技術 HAL及移植要領



主講：Jollen Chen
Email：jollen@jollen.org
Blog：jollen.org/blog

課程日期：2009年10月6日
課程時間：10:00-16:30，共4.5小時
上課地點：台大集思會議中心洛克廳
主辦單位：CTimes

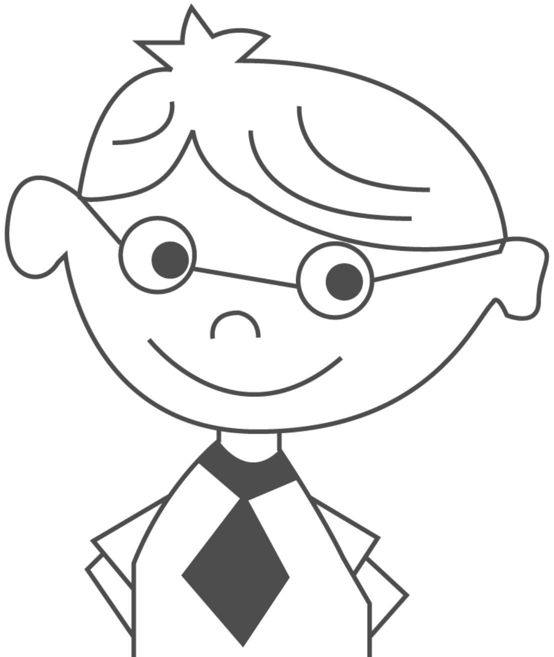
Android 驅動開發關鍵技術：HAL 及移植要領

課程形式： ☑演講(Presentation) □訓練(Training) □實作指導(Lab)

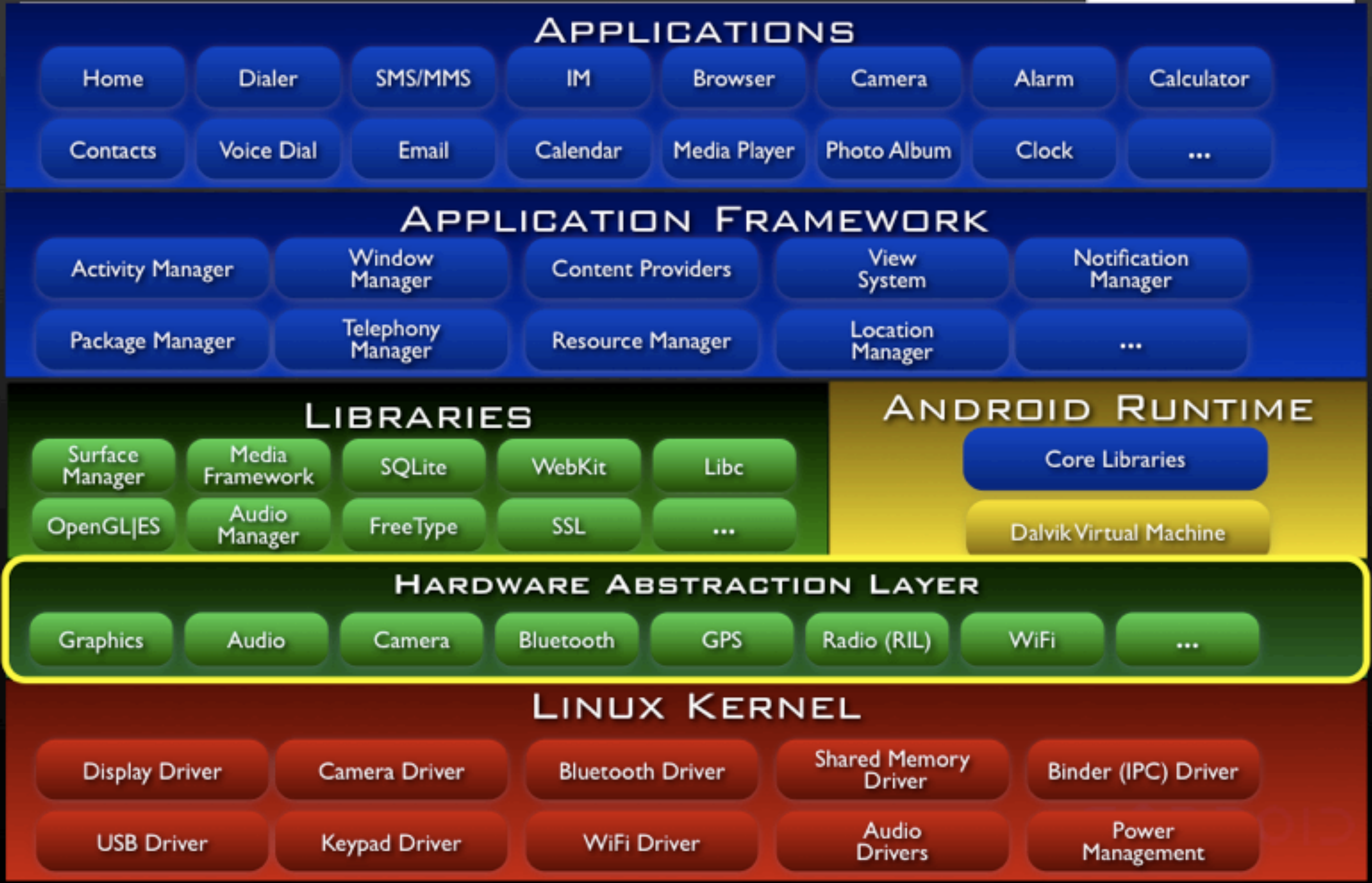
- 本課程由 Jollen’s Consulting 提供
  - http://www.jollen.org/consulting
- 講義電子檔委託仕橙3G教室維護並公佈於 www.moko365.com
- 本課程為演講形式，課程所做操作僅為展示、不做為教學內容

HAL 的架構規劃

- 下圖是 Patrick Brady (Google) 在2008 Google I/O 所發表的演講「Anatomy & Physiology of an Android」中，所提出的 Android HAL 架構圖
- 因應廠商「希望不公開源碼」的要求下，所推出的新觀念，其架構如下圖。雖然 HAL 現在的「抽象程度」還不足，現階段實作還不是全面符合 HAL 的架構規劃



Hardware Abstraction Layer



APPLICATIONS

Home, Dialer, SMS/MMS, IM, Browser, Camera, Alarm, Calculator

Contacts, Voice Dial, Email, Calendar, Media Player, Photo Album, Clock, ...

APPLICATION FRAMEWORK

Activity Manager, Window Manager, Content Providers, View System, Notification Manager

Package Manager, Telephony Manager, Resource Manager, Location Manager, ...

LIBRARIES

Surface Manager, Media Framework, SQLite, WebKit, Libc

OpenGL|ES, Audio Manager, FreeType, SSL, ...

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

HARDWARE ABSTRACTION LAYER

Graphics, Audio, Camera, Bluetooth, GPS, Radio (RIL), WiFi, ...

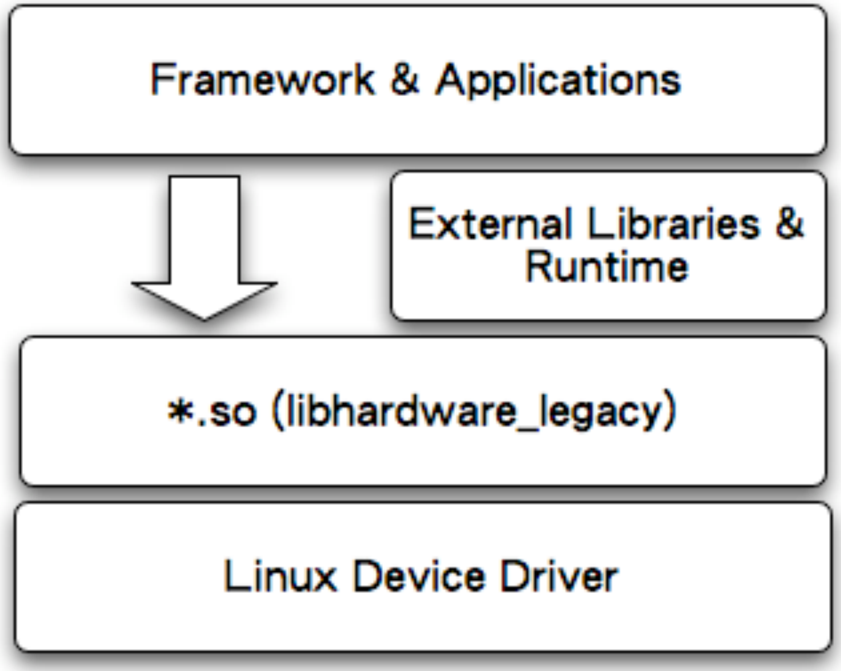
LINUX KERNEL

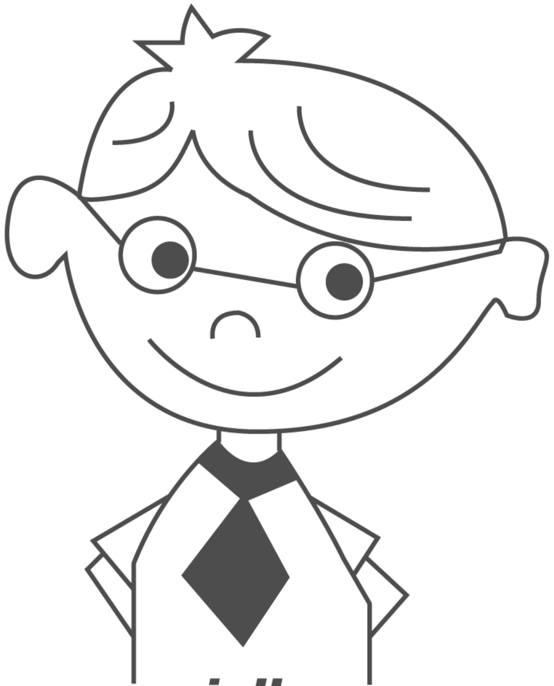
Display Driver, Camera Driver, Bluetooth Driver, Shared Memory Driver, Binder (IPC) Driver

USB Driver, Keypad Driver, WiFi Driver, Audio Drivers, Power Management

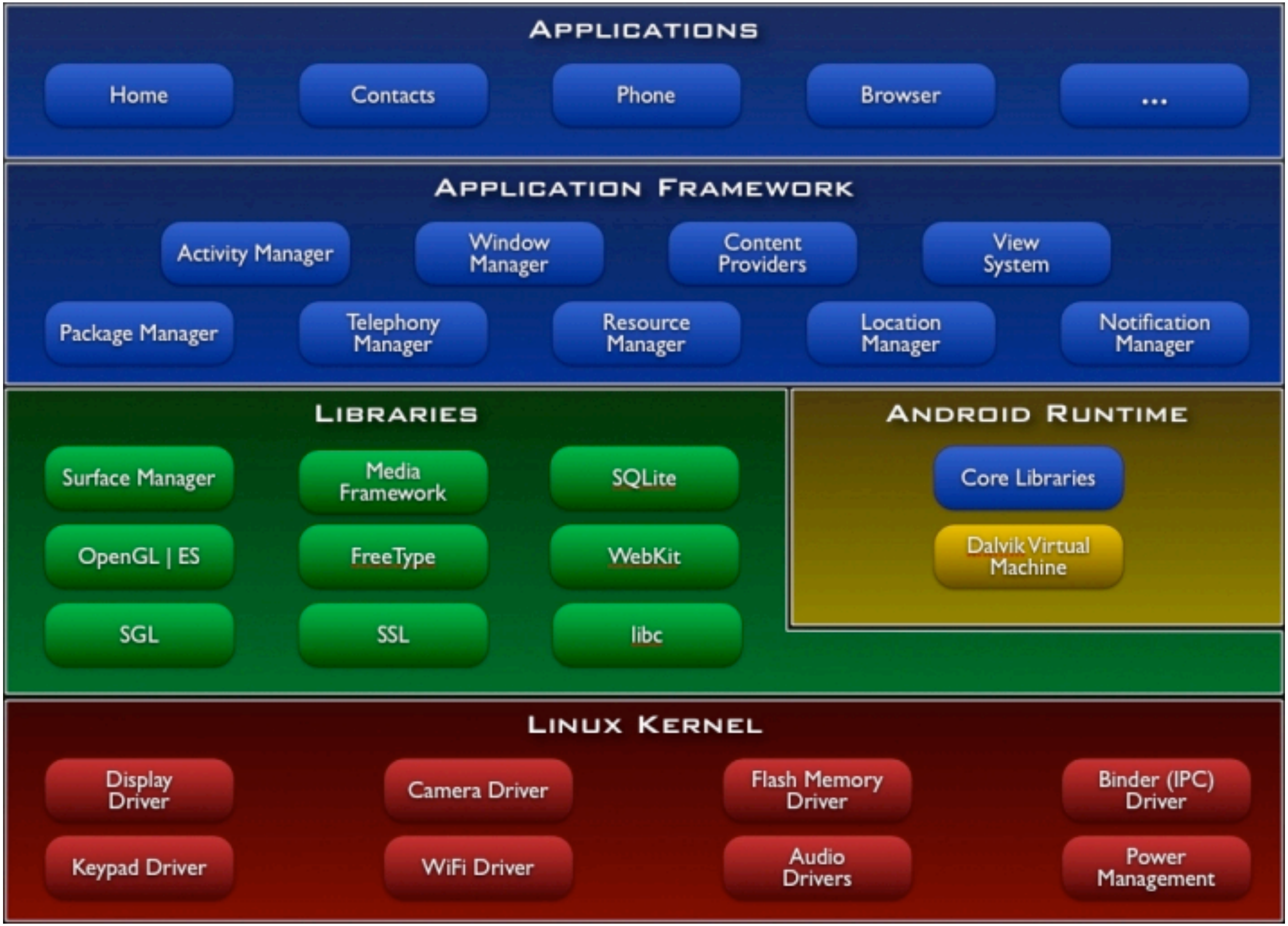
與原始架構並行

- □ 目前實作，仍與原始架構（下頁圖）並行
- □ 驅動程式的移植與開發、仍需要變動 Runtime 的實作





www.jollen.org



Applications

Home
Contacts
Phone
Browser
...

Application Framework

Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager

Libraries

Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc

Android Runtime

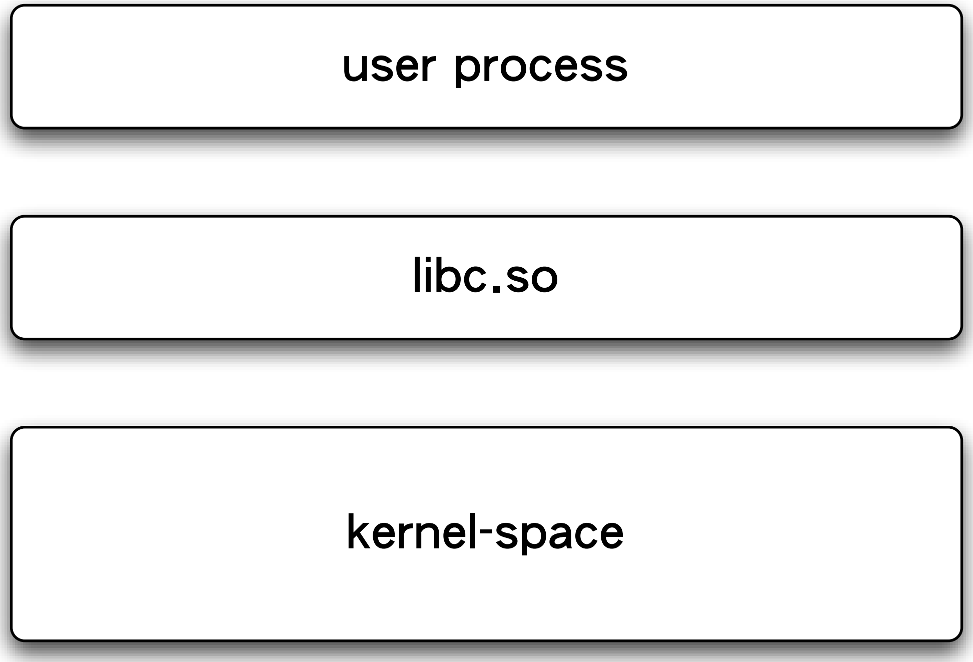
Core Libraries
Dalvik Virtual Machine

Linux Kernel

Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Linux Native Program

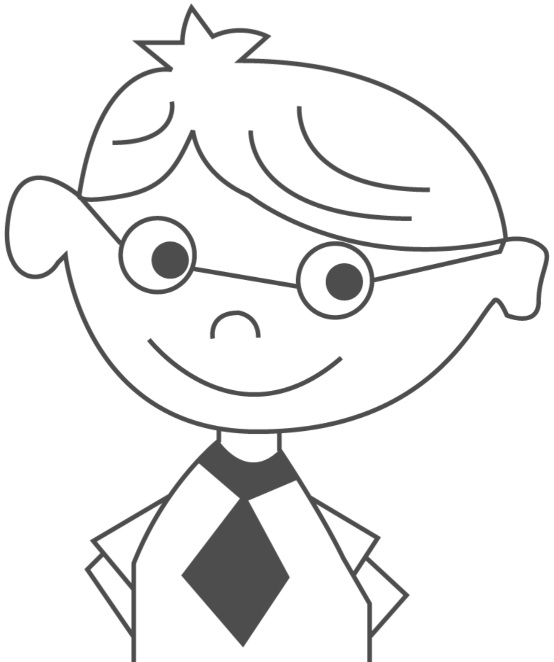
moko365.com



user process

libc.so

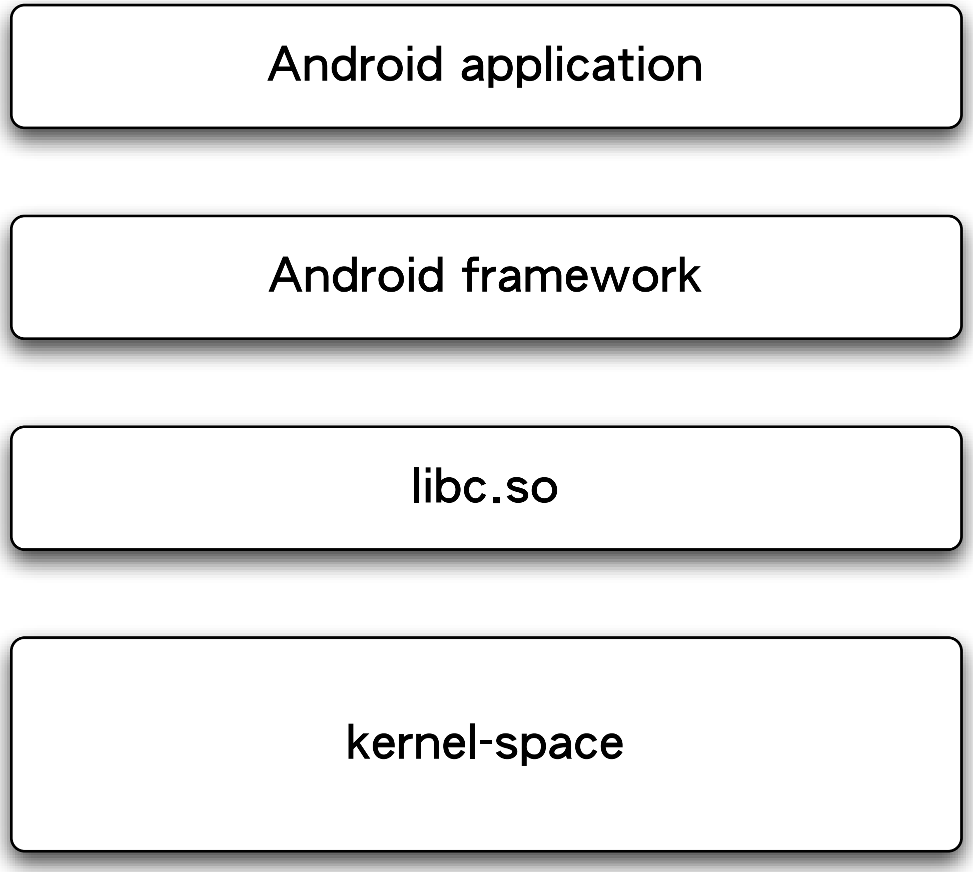
kernel-space



www.jollen.org

Android Program

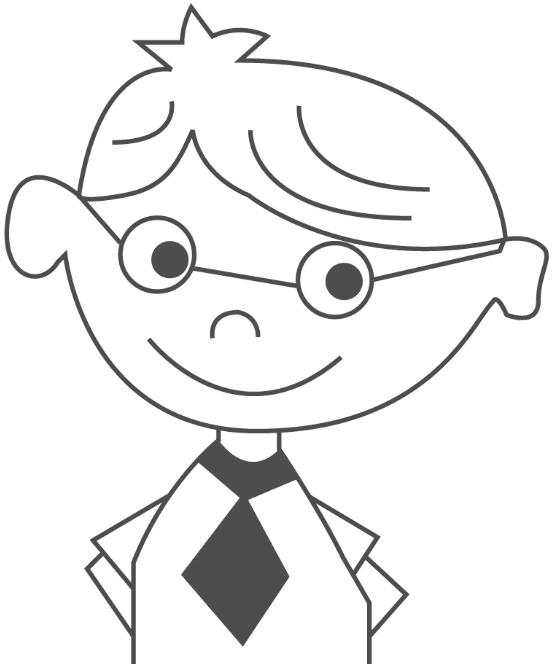
moko365.com


Android application

Android framework

libc.so

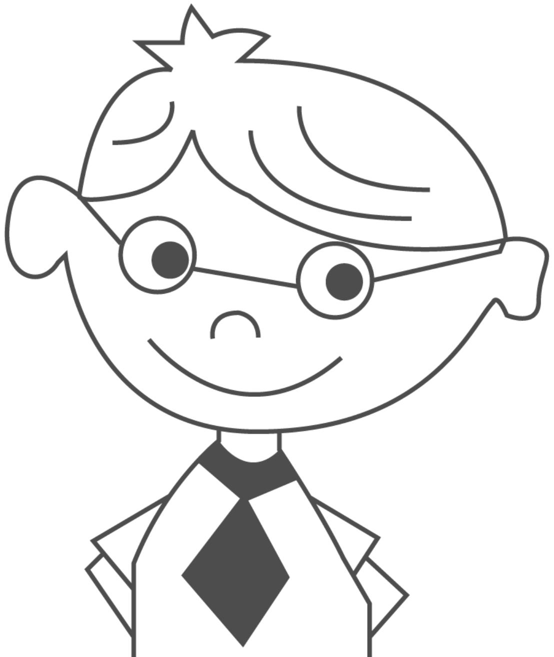
kernel-space


www.jollen.org

Runtime (Core Libraries)

- Sensor Service
- Wifi Service
- XXX Service

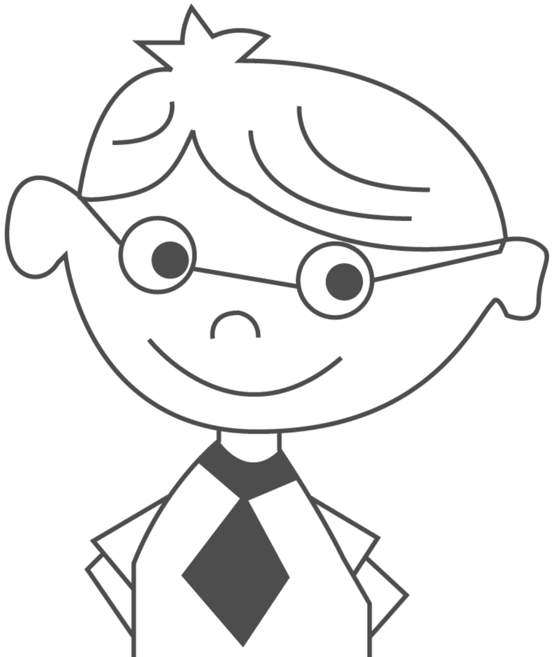
moko365.com



www.jollen.org

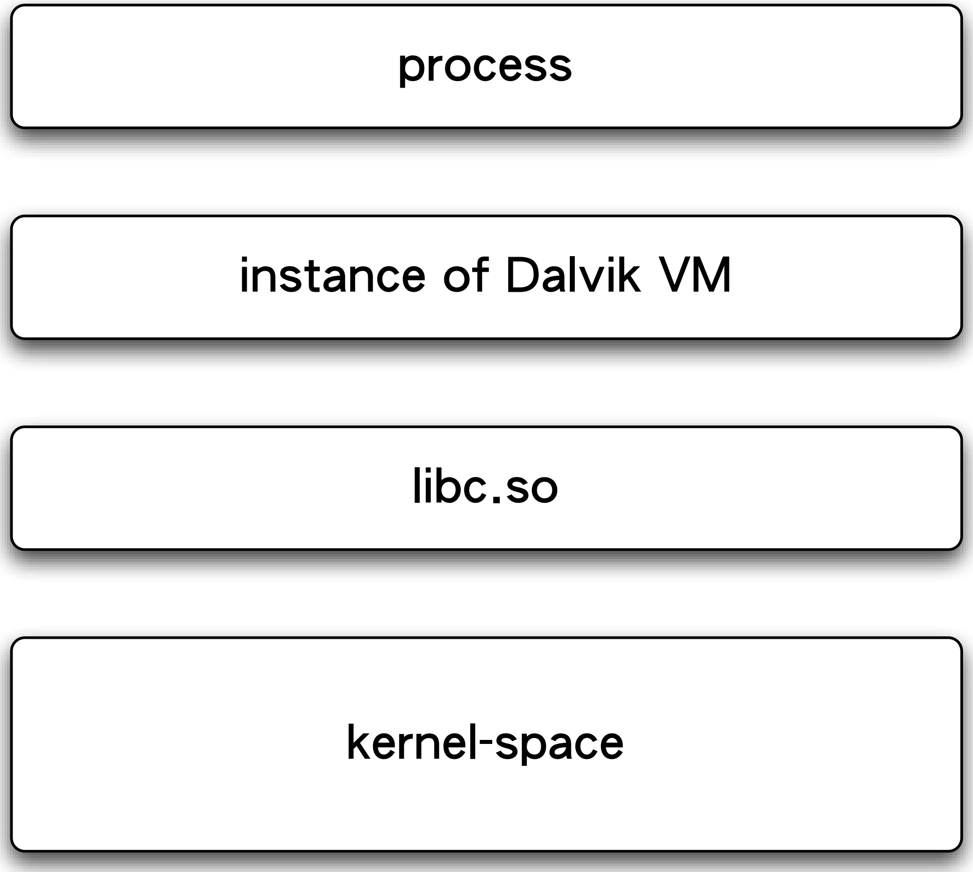
Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

--Android Dev Guide



Android 架構

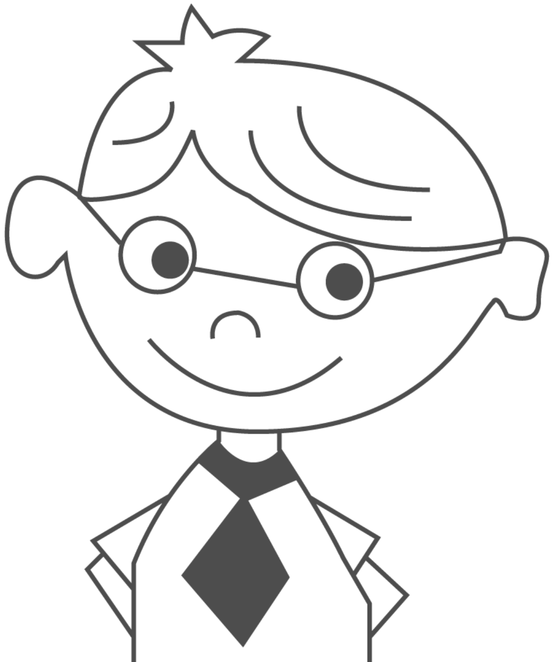
moko365.com


process

instance of Dalvik VM

libc.so

kernel-space

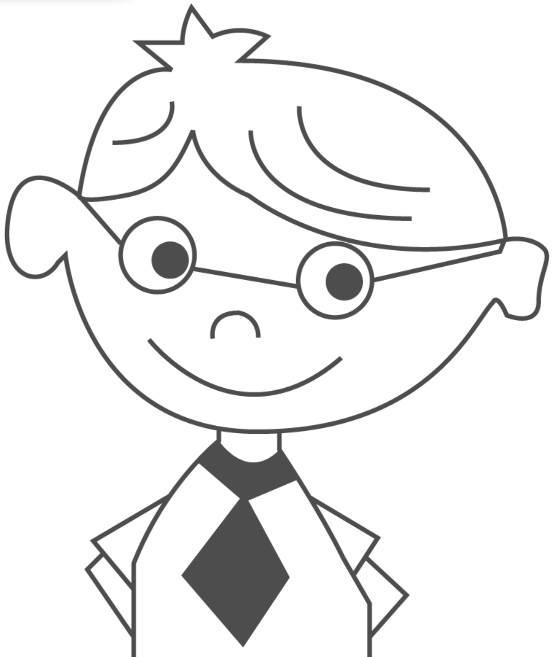


www.jollen.org

預設模式

moko365.com

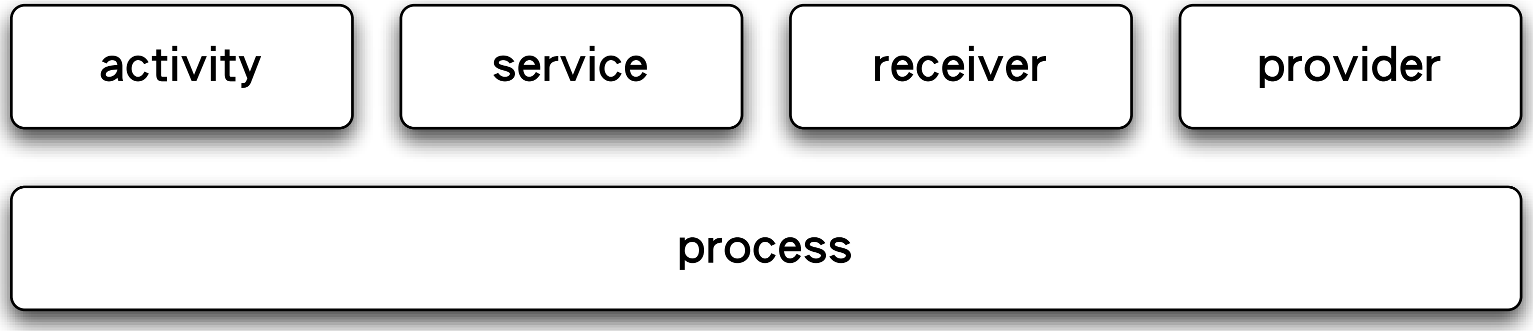
activity	service	receiver	provider
process	process	process	process



www.jollen.org

透過 Manifest 定義

moko365.com



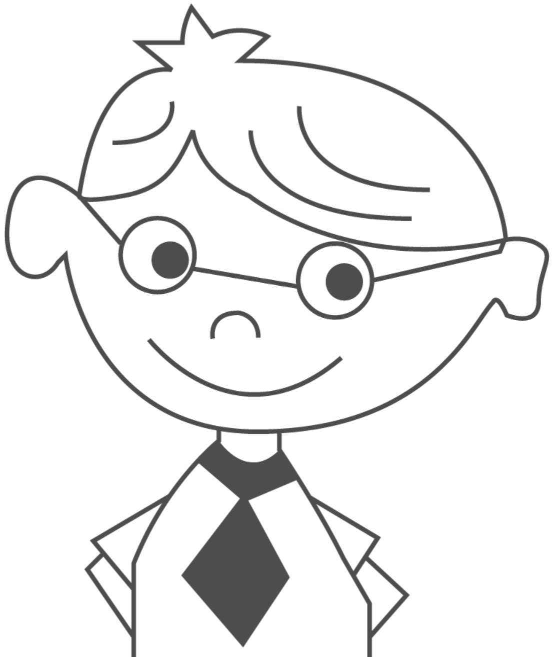
activity

service

receiver

provider

process

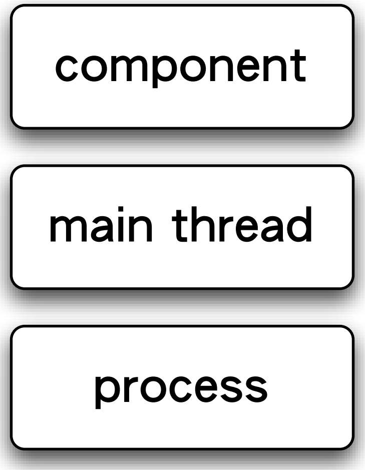


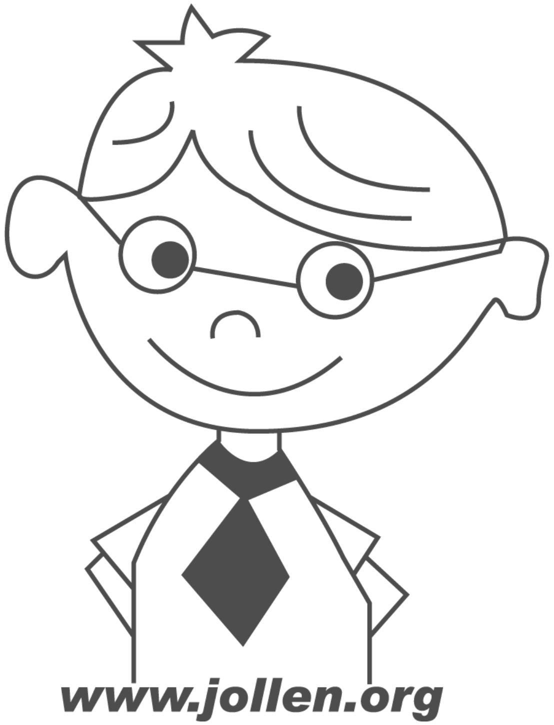
www.jollen.org

Main Thread 觀念

- 每一個 component 一個 process、一個 thread

moko365.com

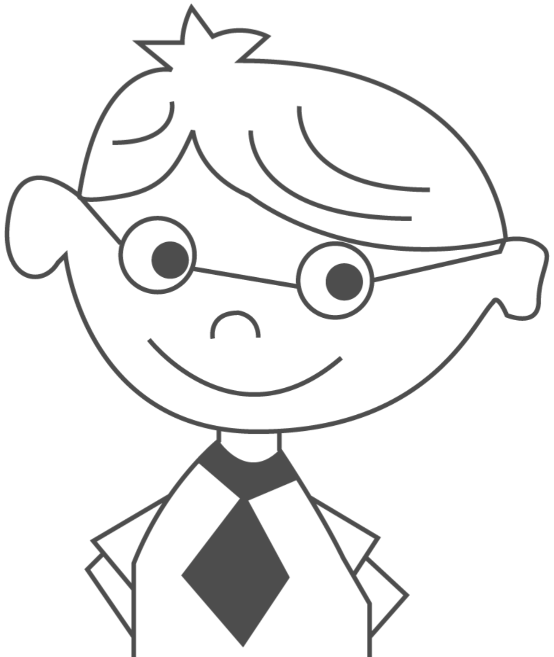




Dalvik VM 特性

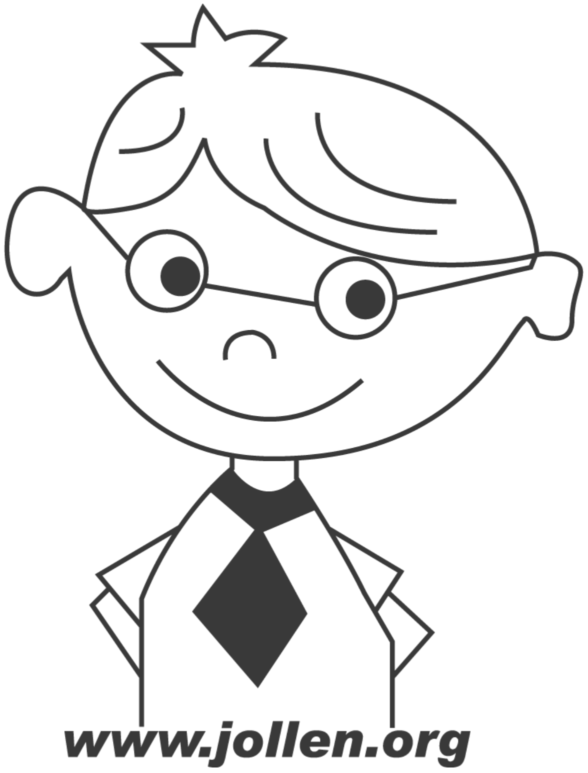
- Zygote 管理所有 Dalvik VM process
- Dalvik VM process 可平行執行
- process 共享 loaded class

moko365.com



www.jollen.org

《Android應用開發與底層技術》Copyright (c) 2009 Jollen’s Consulting 課程開發與提供. www.jollen.org/consulting

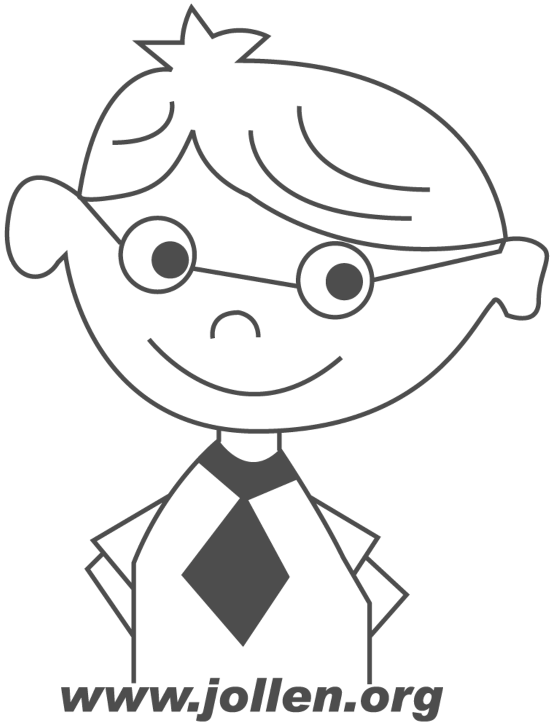


Talk II

Linux Process 模式

- Parent process 與 child process
- Main process 與 threads

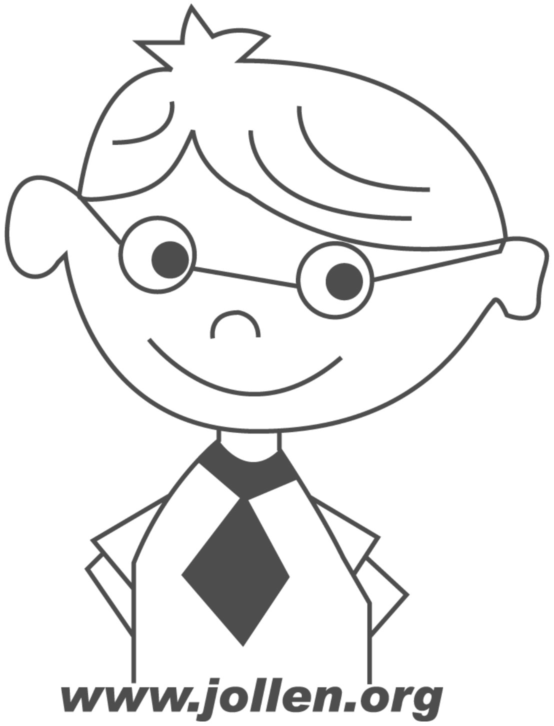
moko365.com



Linux IPC

- shared memory
- mmap()
- message queue
- etc.

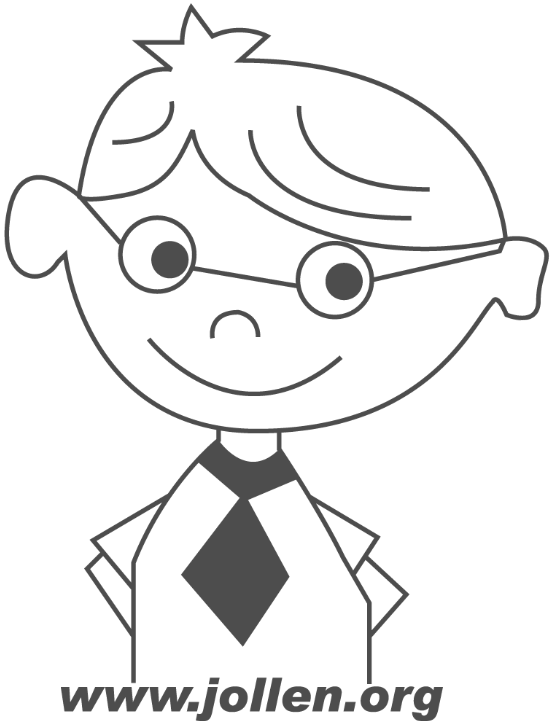
moko365.com



Blocking Operations

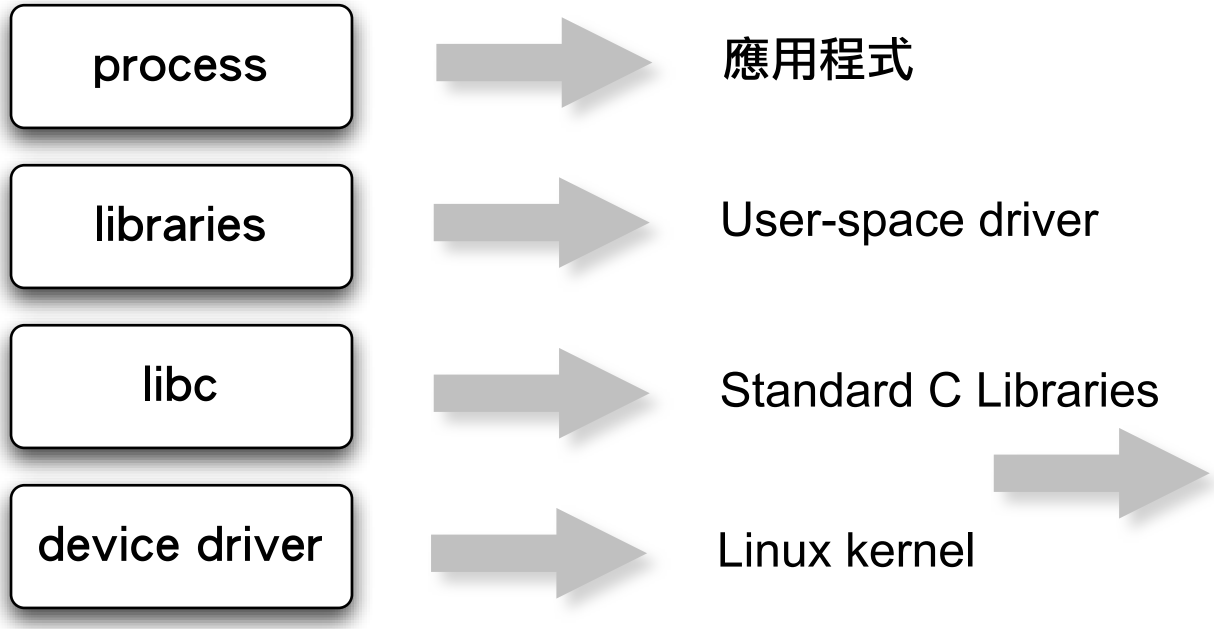
moko365.com

- 「Separate threads are not created for each instance」
- 回應以下動作的 methods 不應該在 main thread 裡執行
  - long operation
  - blocking operation
- 分離 (spawn) 出新的 thread 來處理



◆Linux 驅動程式的技術架構

moko365.com



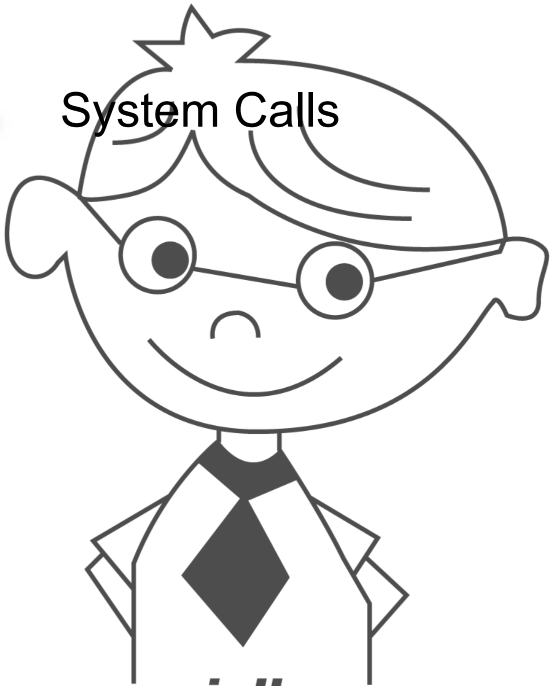
process → 應用程式

libraries → User-space driver

libc → Standard C Libraries

device driver → Linux kernel

→ System Calls

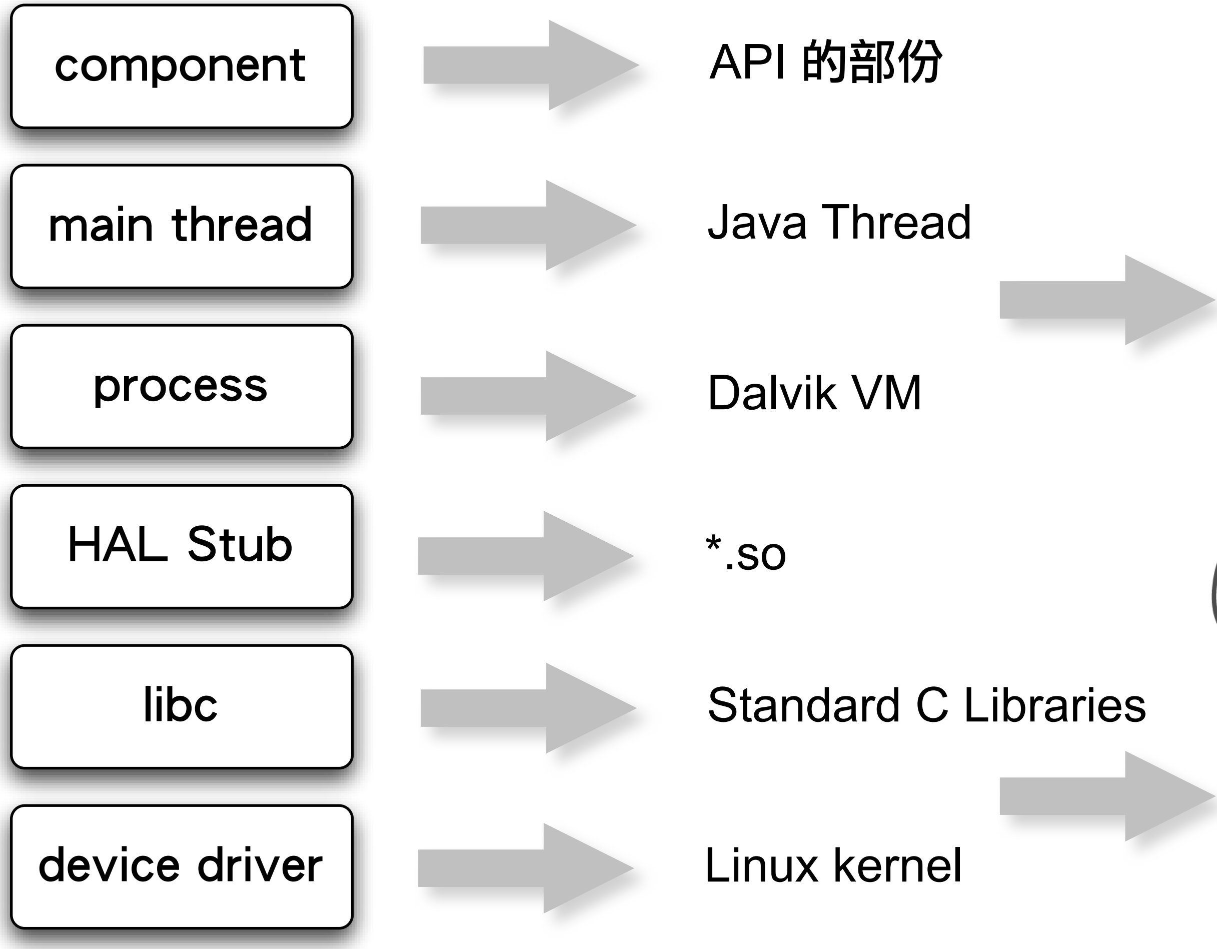


www.jollen.org

* 製圖: *Jollen's Consulting, for update see jollen.org/consulting*

◆Android 驅動程式的技術架構

moko365.com



component → API 的部份

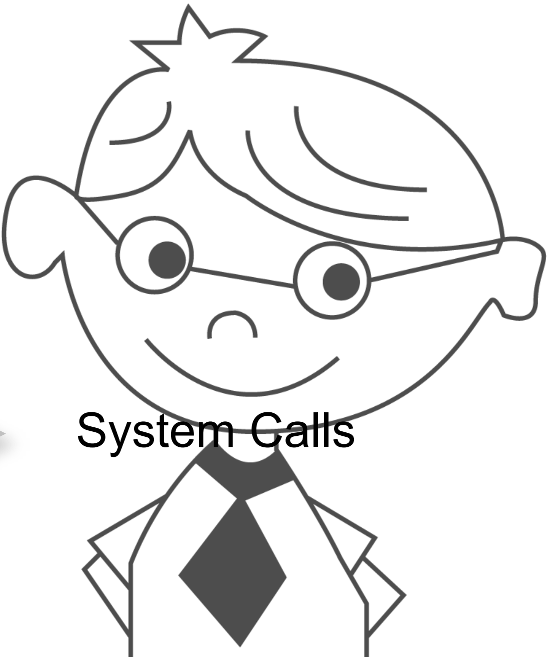
main thread → Java Thread → JNI

process → Dalvik VM

HAL Stub → *.so

libc → Standard C Libraries → System Calls

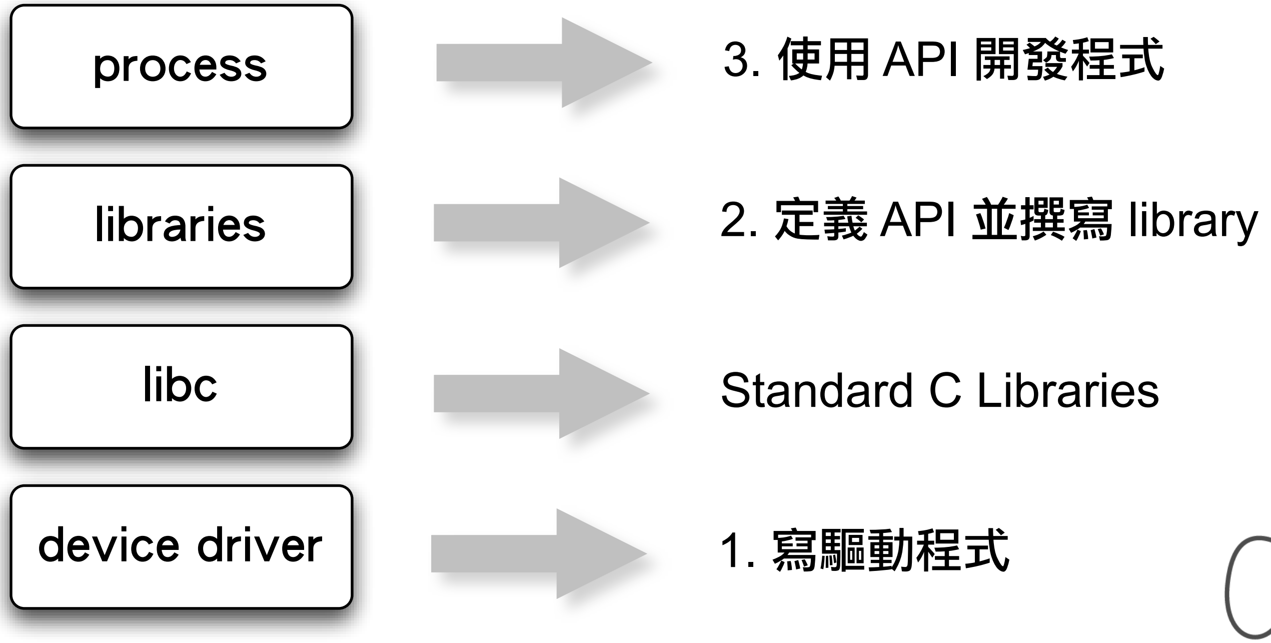
device driver → Linux kernel



www.jollen.org

* 製圖: *Jollen’s Consulting, for update see jollen.org/consulting*

◆Android 驅動程式的開發流程



process → 3. 使用 API 開發程式

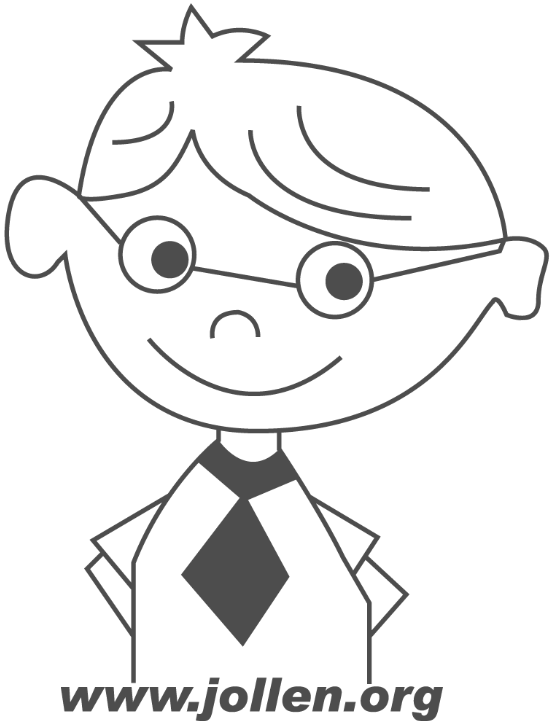
libraries → 2. 定義 API 並撰寫 library

libc → Standard C Libraries

device driver → 1. 寫驅動程式

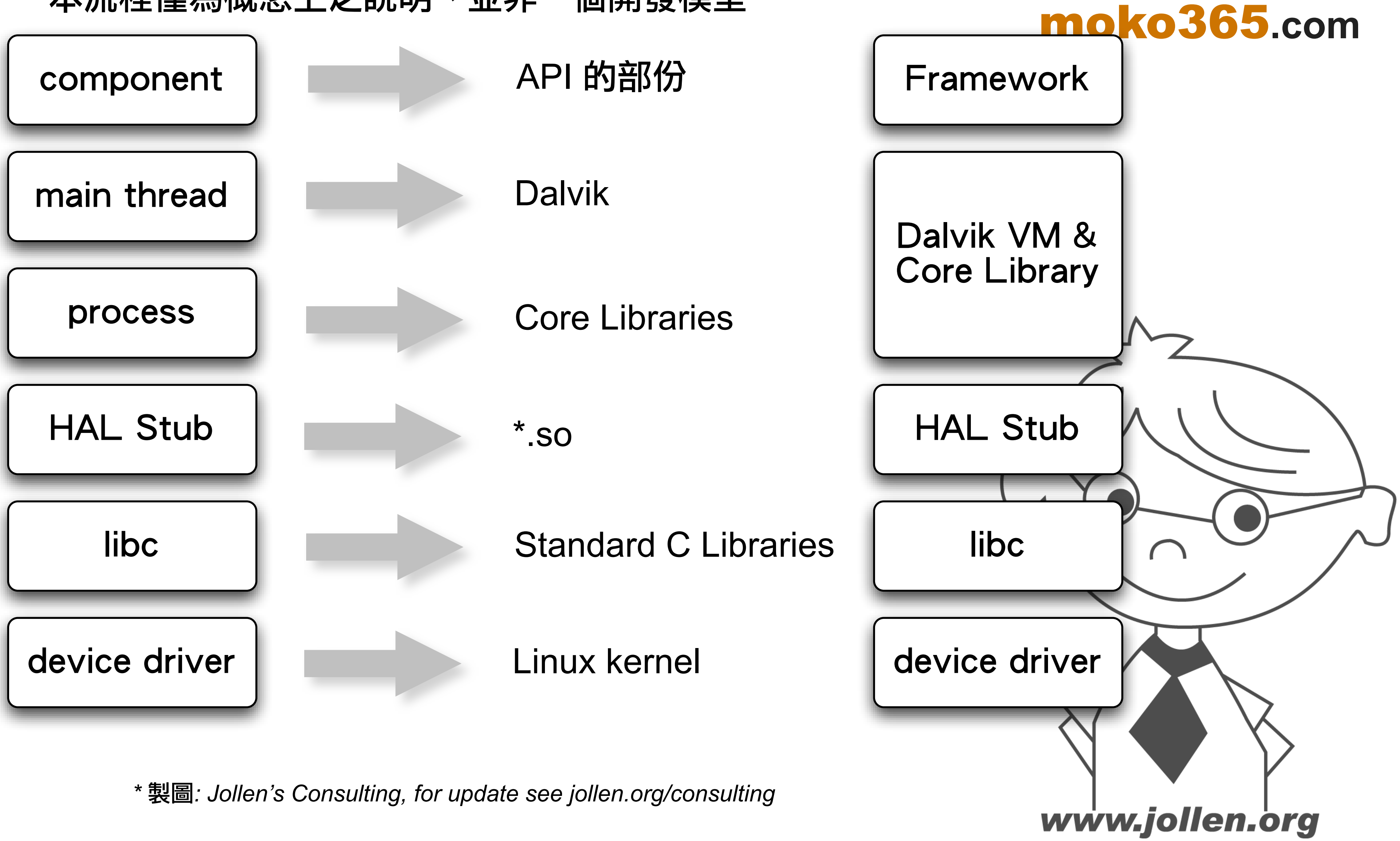
* 本流程僅為概念上之說明、並非一個開發模型

* 製圖: *Jollen’s Consulting, for update see jollen.org/consulting*



◆Android 驅動程式的技術架構

* 本流程僅為概念上之說明、並非一個開發模型

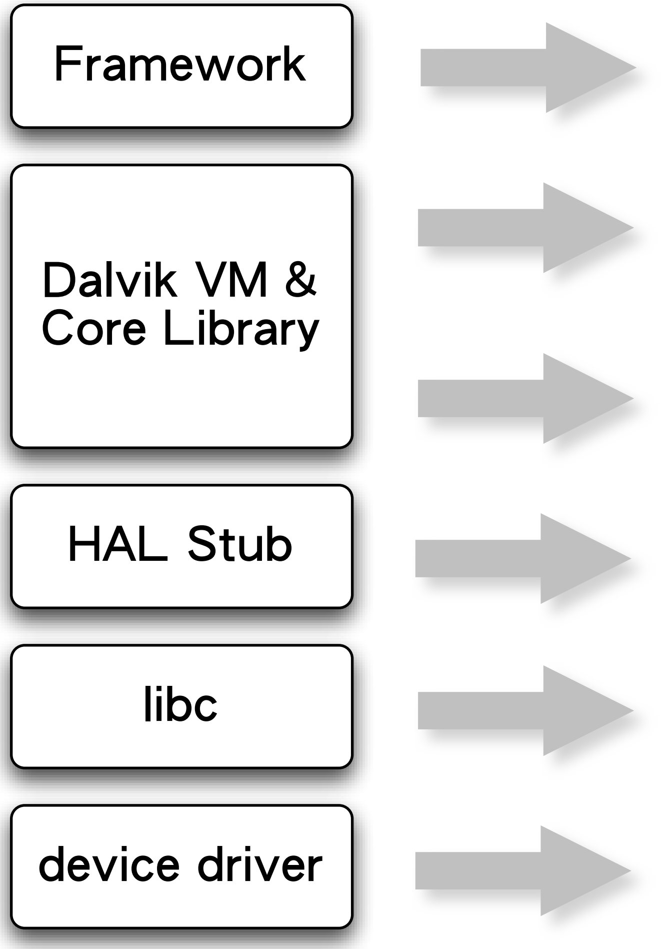


component	API 的部份	Framework
main thread	Dalvik	Dalvik VM & Core Library
process	Core Libraries	
HAL Stub	*.so	HAL Stub
libc	Standard C Libraries	libc
device driver	Linux kernel	device driver

* 製圖: *Jollen's Consulting, for update see jollen.org/consulting*

◆Android 驅動程式的開發流程

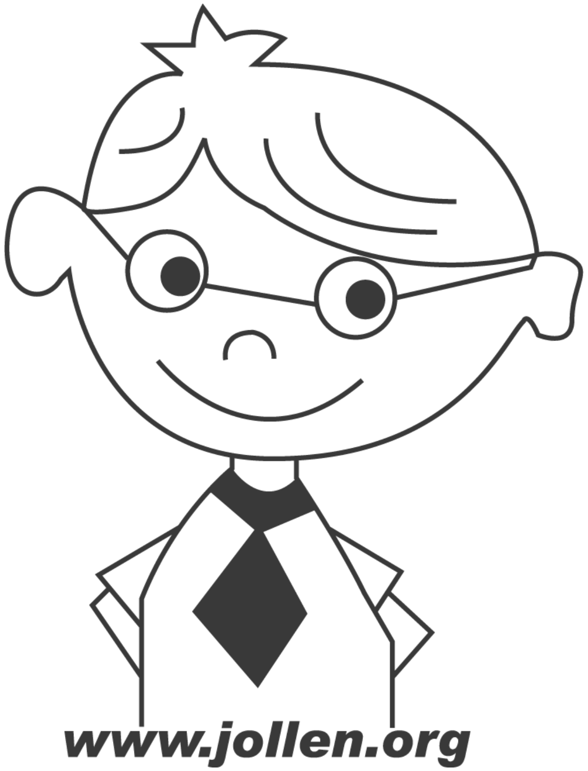
moko365.com



Framework	2. 定義 API 與 native function
Dalvik VM & Core Library	3. 定義 JNI method table 5. 實作 core libraries
HAL Stub	4. 定義 callback functions 與 supporting API
libc	Standard C Libraries
device driver	1. 寫驅動程式

* 製圖: *Jollen’s Consulting, for update see jollen.org/consulting*

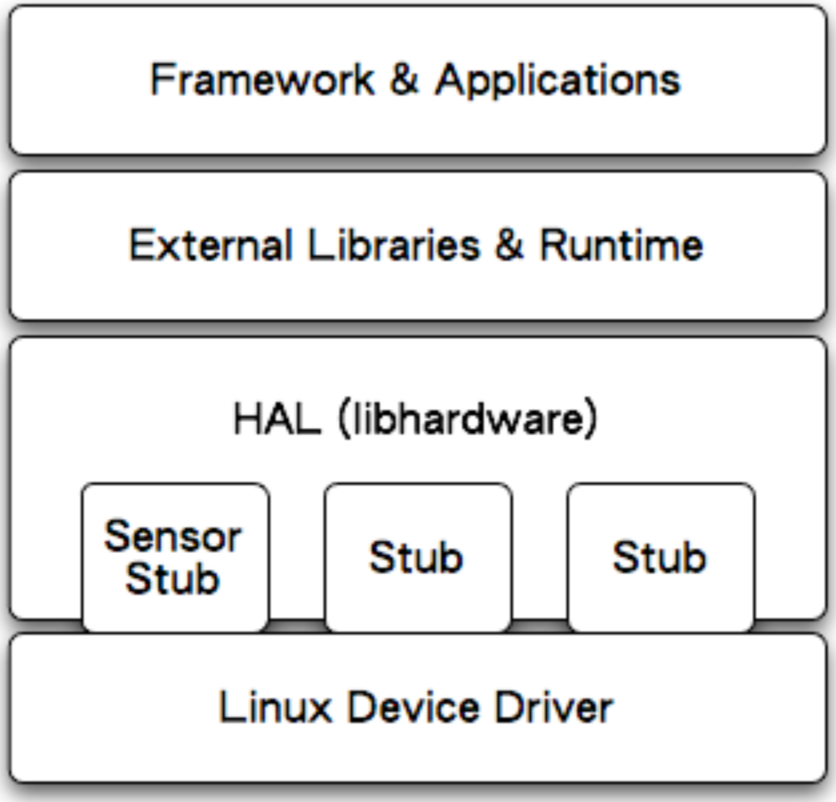
www.jollen.org

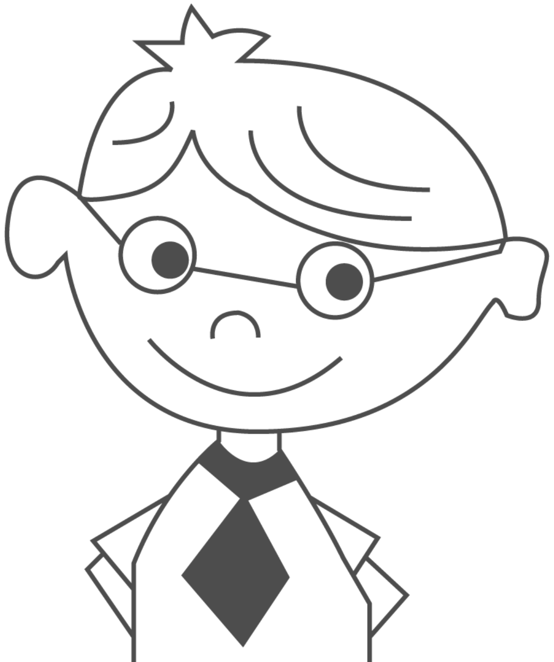


Talk III、IV

透過 Runtime 取得 Stub ops

- 現行的 HAL 實作、抽象程度不高
- 應用程式透過 Runtime (Service) 來取得 stub 的 operations

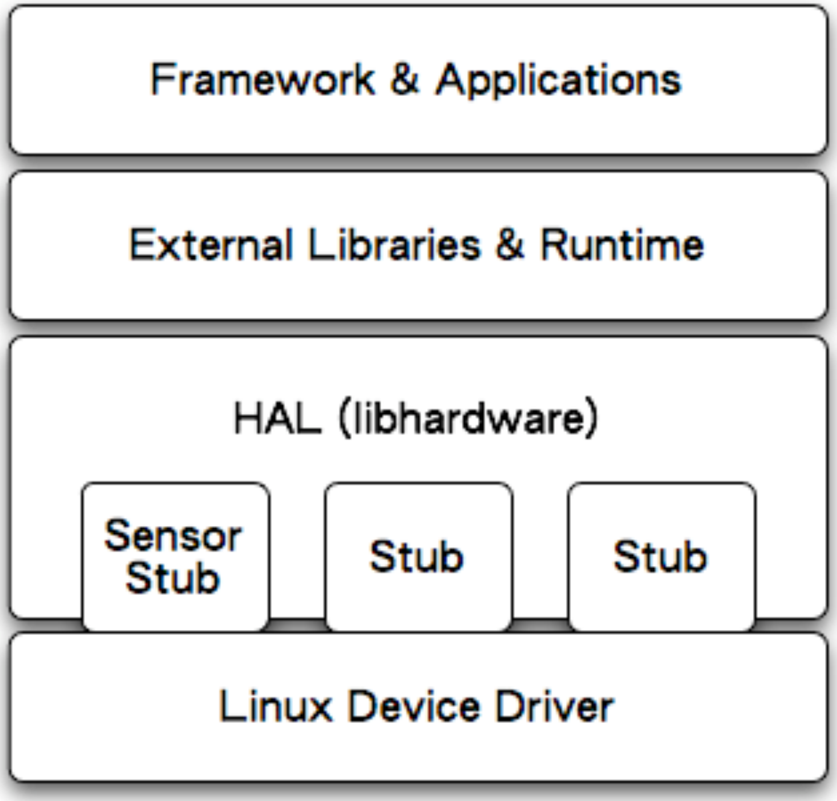


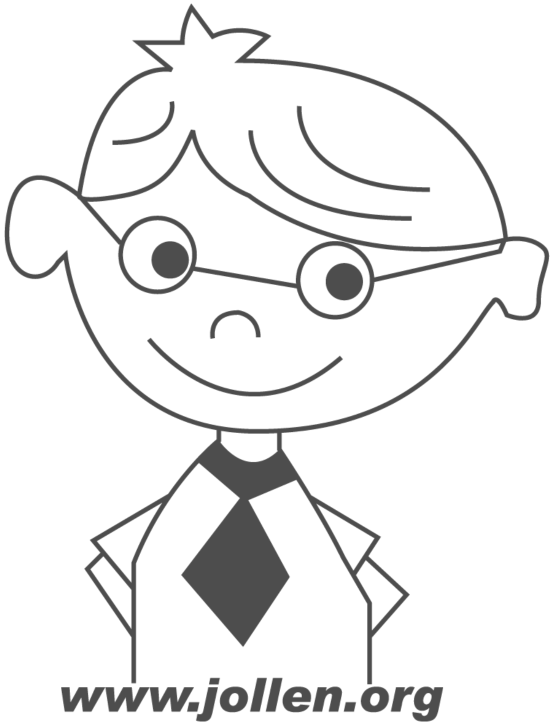


仍需變動框架實作

- 現行的 HAL 在移植驅動程式時，可能需要變動 Runtime (Service) 部份
- 暫無法達到「只做 HAL stub」不變動 framework 的移植理想

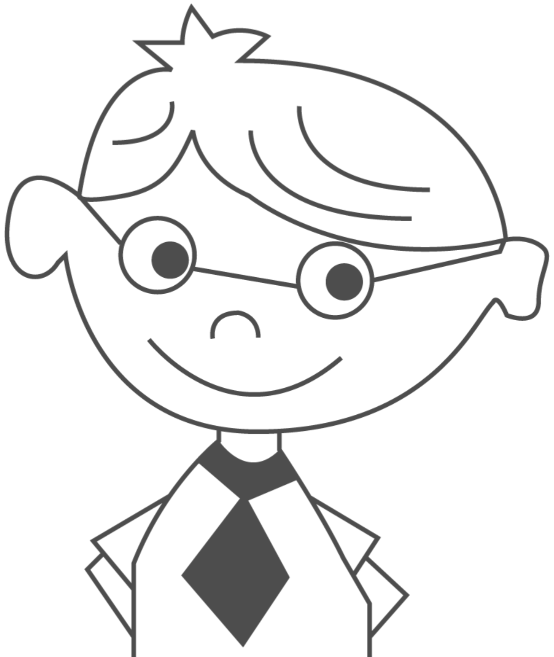
moko365.com



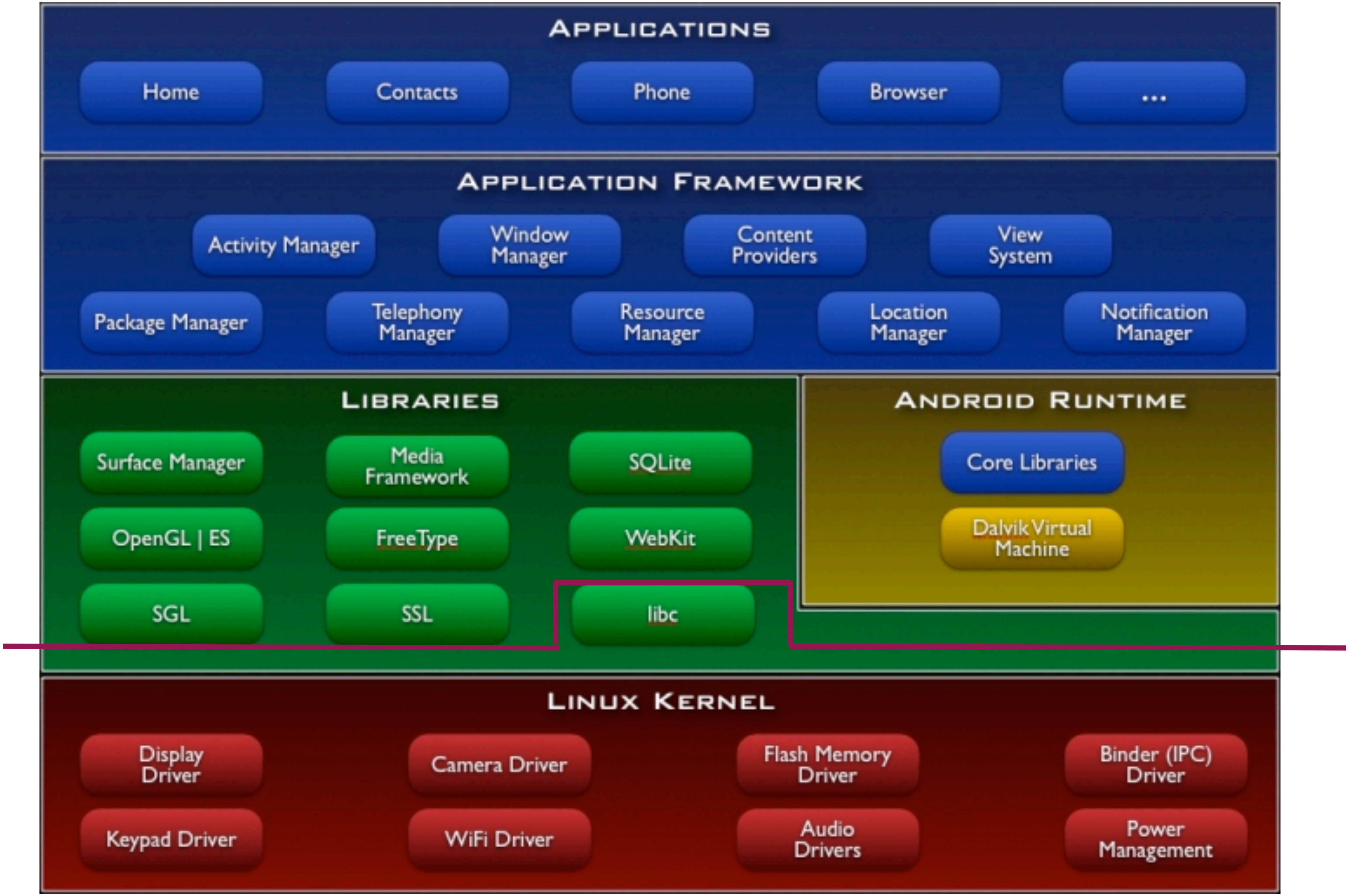


實例說明：Sensor Service

- 以 Sensor Service 說明 HAL 與驅動程式移植的實際做法
- 目前的 Android、所有的 Service 暫時還不是統一做法
- HAL Stub 需要 libc (system call)
- 仍需要 case-by-case 研究
  - 例如：Camera 的 HAL



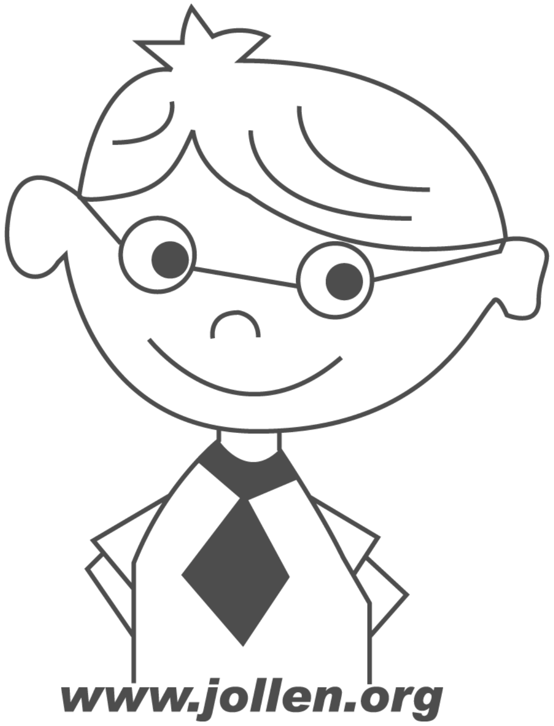
www.jollen.org


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

libhardware 的角色

moko365.com

- libhardware 讓驅動程式開發者撰寫 HAL module (HAL stub)
- Dalvik VM process 使用 libhardware 取得 HAL stub 的 callback ops
- 透過 callback functions 與驅動程式溝通

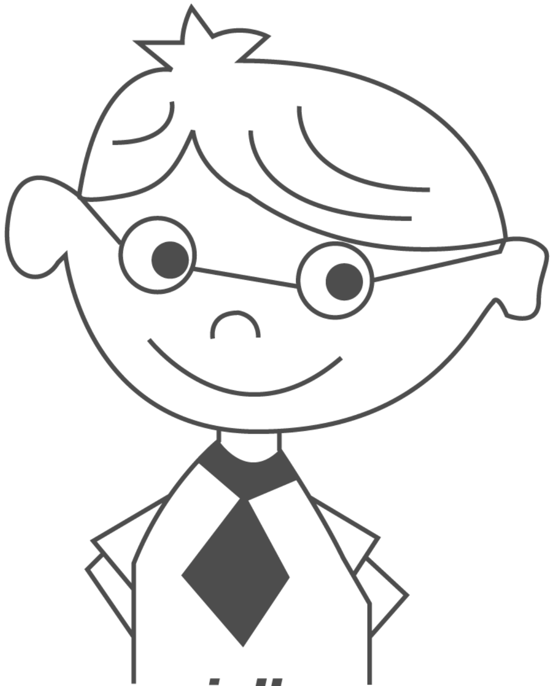


◆如何取得 HAL Module

moko365.com

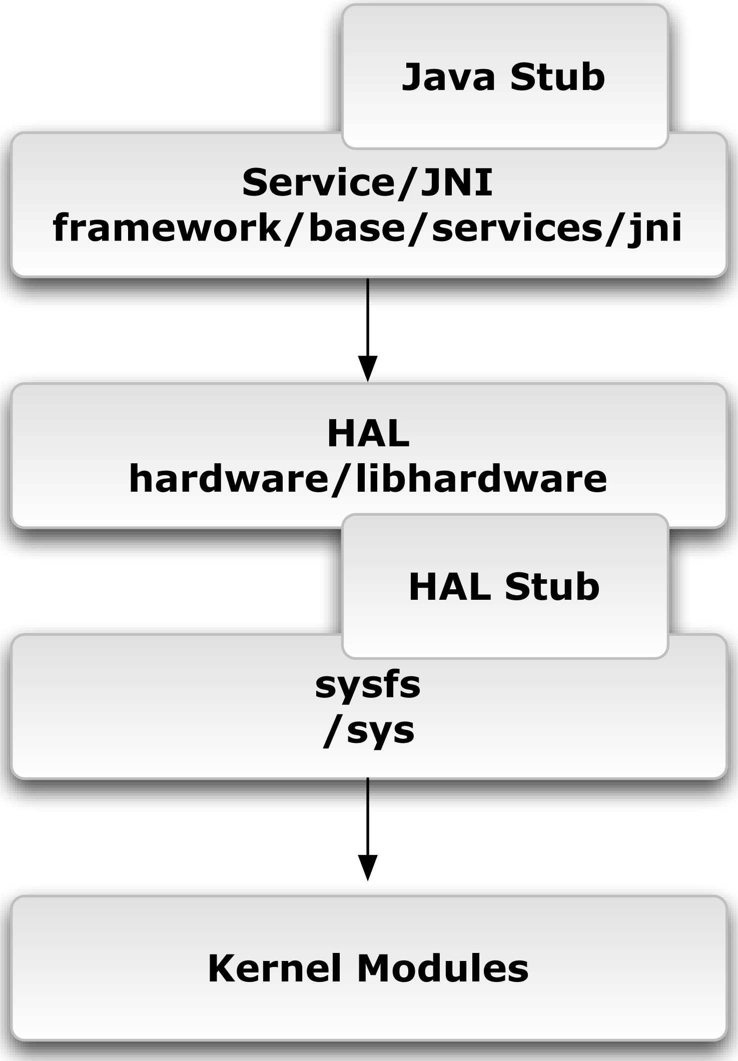
```
int hw_get_module(const char *id, const struct hw_module_t **module)
```

- id: HAL Module ID
- *module: HAL Module Operations

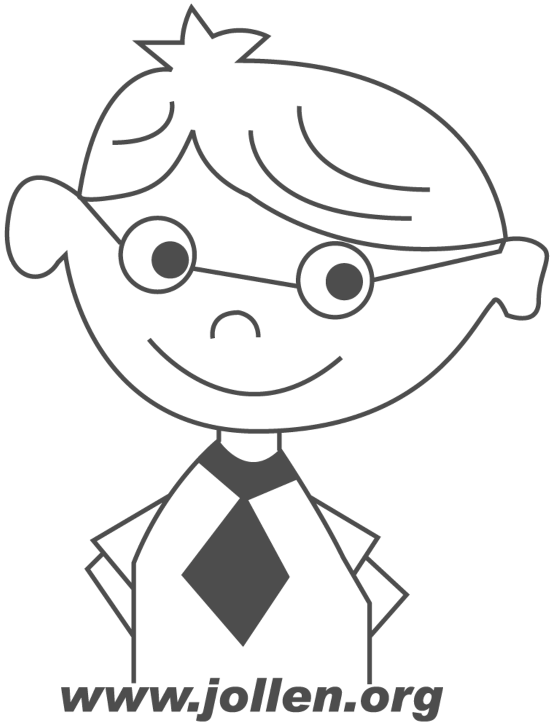


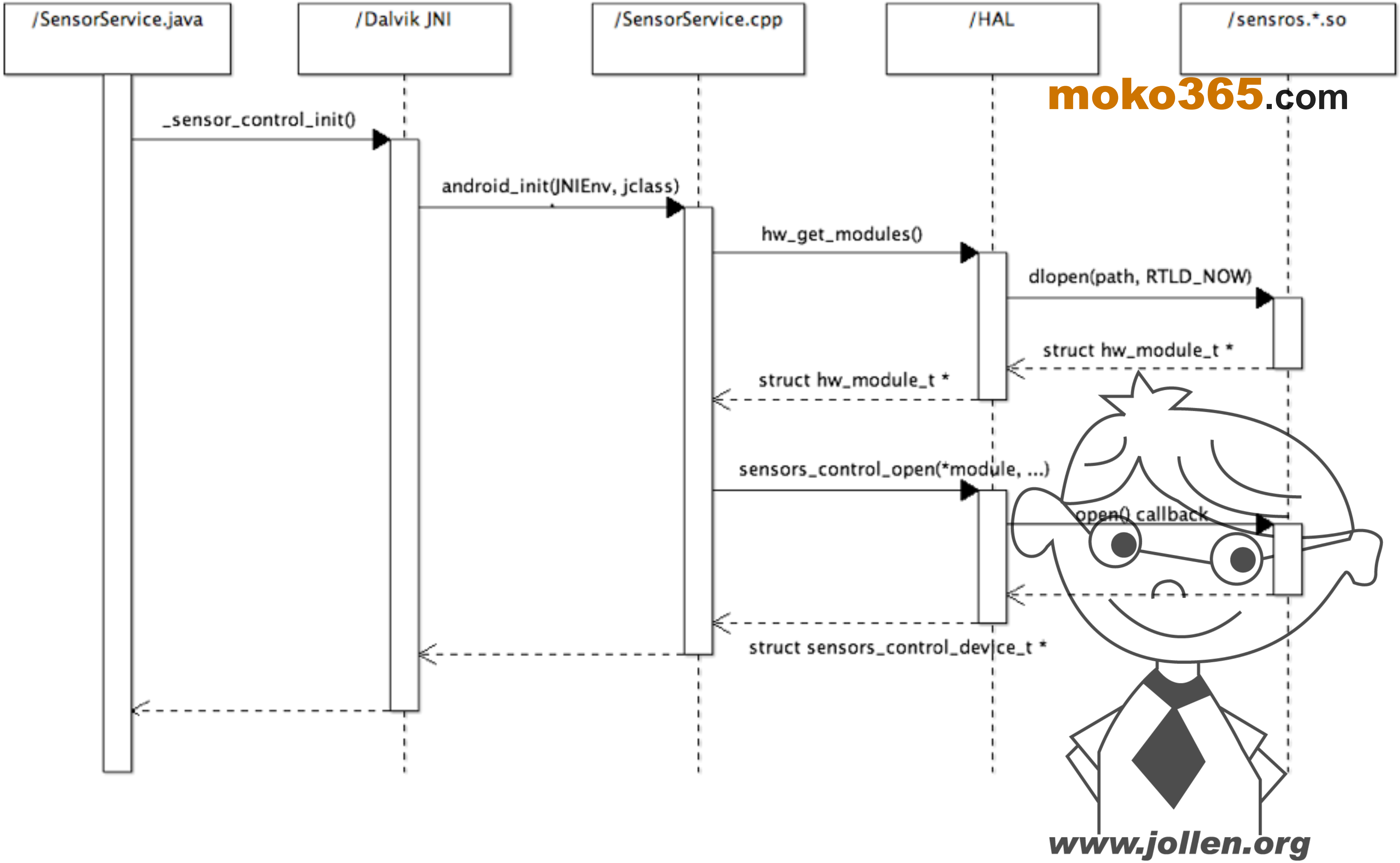
www.jollen.org

HAL Stub 的用途



moko365.com

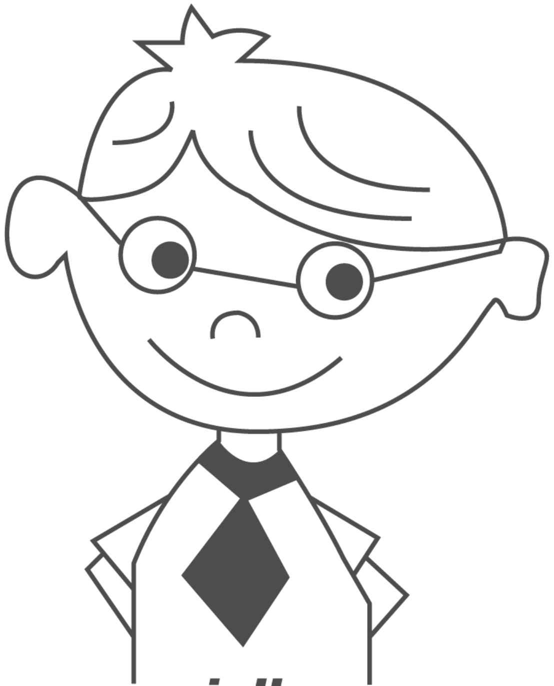



/SensorService.java
/Dalvik JNI
/SensorService.cpp
/HAL
/sensros.*.so
_sensor_control_init()
android_init(JNIEnv, jclass)
hw_get_modules()
dlopen(path, RTLD_NOW)
struct hw_module_t *
struct hw_module_t *
sensors_control_open(*module, ...)
open() callback
struct sensors_control_device_t *

◆Step1: 定義 “init” native function

moko365.com

```
class SensorService extends ISensorService.Stub {
    ...
    private static native int _sensors_control_init();
    private static native ParcelFileDescriptor _sensors_control_open();
    private static native boolean _sensors_control_activate(int sensor, boolean activate);
    private static native int _sensors_control_set_delay(int ms);
}
```



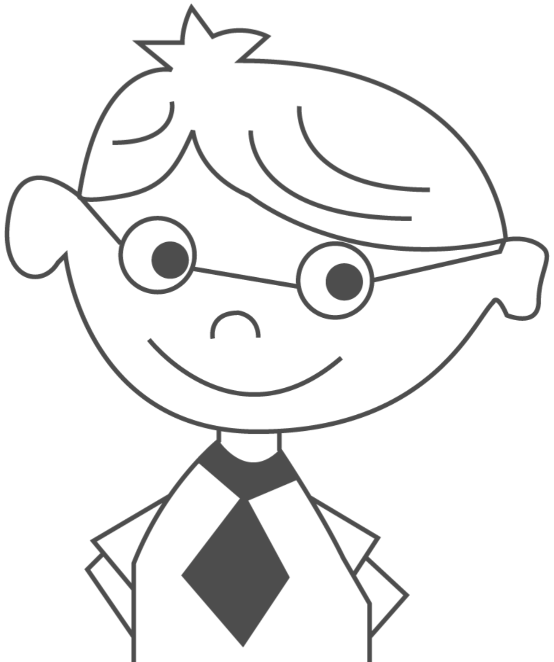
at framework/base/services/java/SensorService.java

www.jollen.org

◆Step2: 建構子裡呼叫 init function

moko365.com

```
class SensorService extends ISensorService.Stub {
    ...
    public SensorService(Context context) {
        if (localLOGV) Log.d(TAG, "SensorService startup");
        _sensors_control_init();
        mNotificationManager = (NotificationManager)context.getSystemService
(Context.NOTIFICATION_SERVICE);
        mContext = context;
    }
   ...
}
```



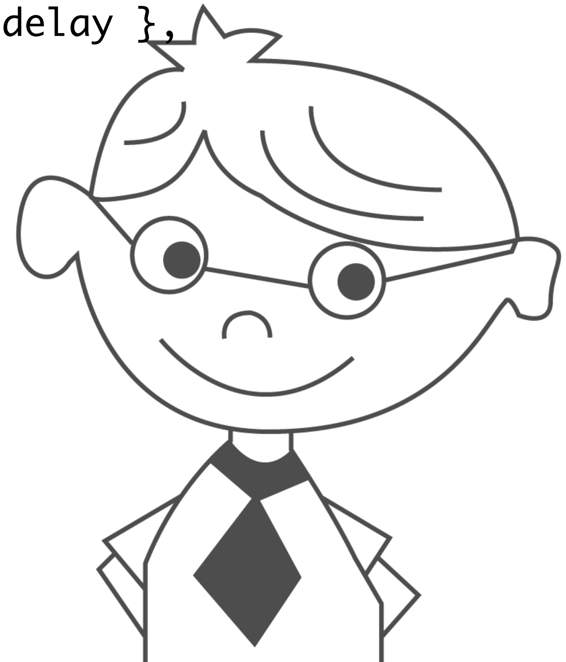
at framework/base/services/java/SensorService.java

www.jollen.org

◆JNI Method Table 定義

moko365.com

```
static JNINativeMethod gMethods[] = {
    {"_sensors_control_init",     "()I",   (void*) android_init },
    {"_sensors_control_open",     "()Landroid/os/ParcelFileDescriptor;",
                                           (void*) android_open },
    {"_sensors_control_activate", "(IZ)Z", (void*) android_activate },
    {"_sensors_control_wake",     "()I", (void*) android_data_wake },
    {"_sensors_control_set_delay","(I)I", (void*) android_set_delay },
};
```



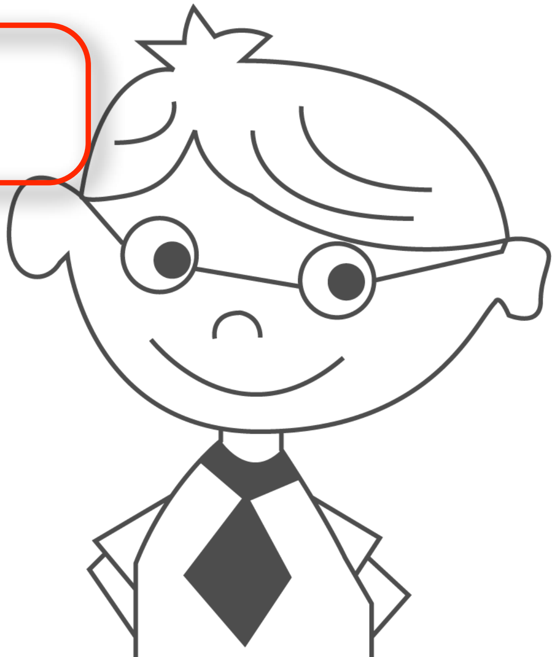
at framework/base/services/jni/com_android_server_SensorService.cpp

www.jollen.org

◆Step 3: 取得 HAL stub 的 callbacks

moko365.com

```
static jint
android_init(JNIEnv *env, jclass clazz)
{
    sensors_module_t* module;
    if (hw_get_module(SENSORS_HARDWARE_MODULE_ID, (const hw_module_t**)&module) == 0) {
        if (sensors_control_open(&module->common, &sSensorDevice) == 0) {
            const struct sensor_t* list;
            int count = module->get_sensors_list(module, &list);
            return count;
        }
    }
    return 0;
}
```

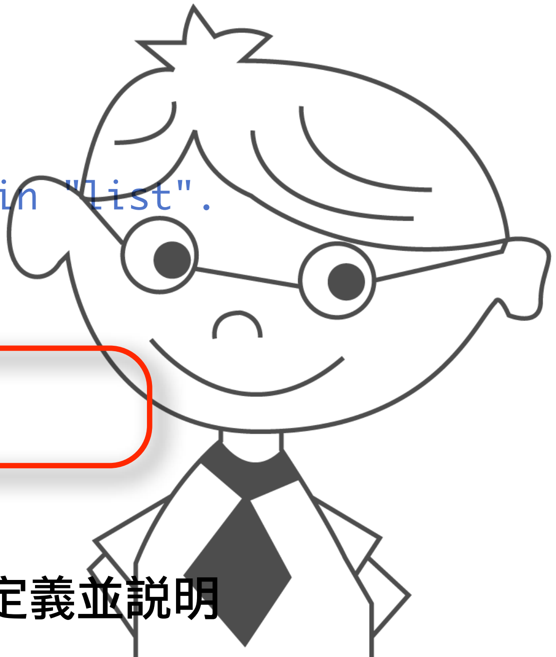


```
#define SENSORS_MODULE_ID “sensors”
```

www.jollen.org

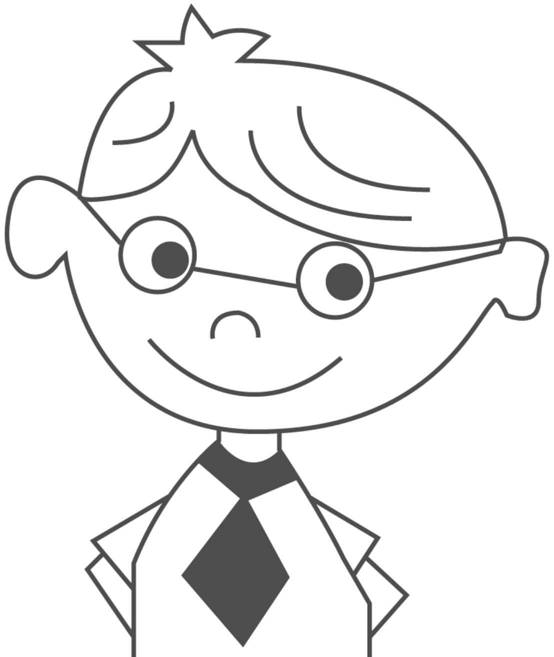
◆ Step 4: 裡 HAL stub 裡定義 struct hw_module_t 的 wrapper

```
/**
 * Every hardware module must have a data structure named HAL_MODULE_INFO_SYM
 * and the fields of this data structure must begin with hw_module_t
 * followed by module specific information.
 */
struct sensors_module_t {
    struct hw_module_t common;

    /**
     * Enumerate all available sensors. The list is returned in "list".
     * @return number of sensors in the list
     */
    int (*get_sensors_list)(struct sensors_module_t* module,
            struct sensor_t const** list);
};
```



supporting API
由 HAL stub 開發者定義並說明

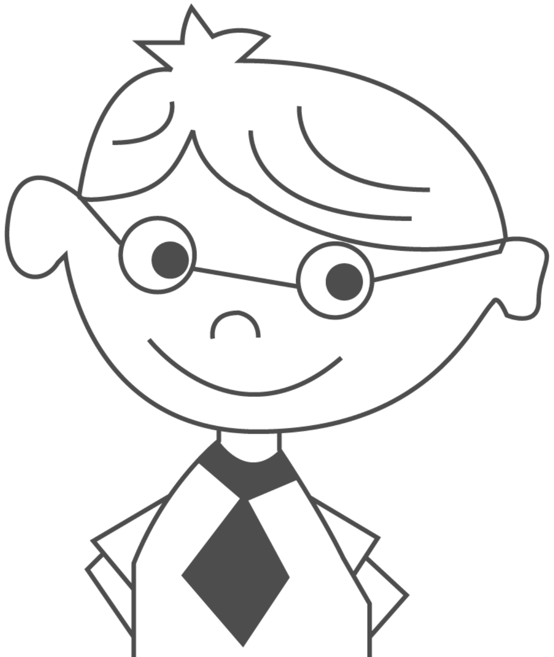
```
struct hw_module_t {
    /** tag must be initialized to HARDWARE_MODULE_TAG */
    uint32_t tag;

    /** major version number for the module */
    uint16_t version_major;

    /** minor version number of the module */
    uint16_t version_minor;

    /** Identifier of module */
    const char *id;

    /** Name of this module */
    const char *name;

    /** Author/owner/implementor of the module */
    const char *author;

    /** Modules methods */
    struct hw_module_methods_t* methods;

    /** padding to 128 bytes, reserved for future use */
    uint32_t reserved[32-6];
};
```



◆Step 5: 實作 HAL stub 的 callbacks

moko365.com

```
struct hw_module_methods_t {
    /** Open a specific device */
    int (*open)(const struct hw_module_t* module, const char* id,
            struct hw_device_t** device);
};
```

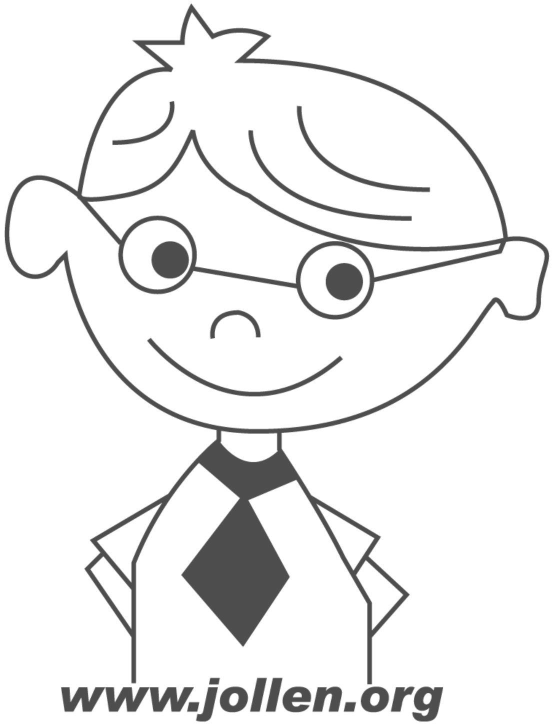


www.jollen.org

◆Step 6: callback HAL stub

moko365.com

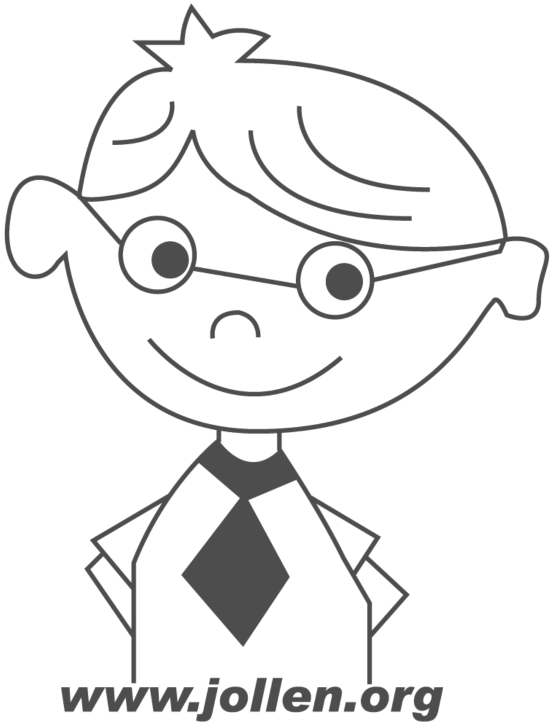
```
static inline int sensors_control_open(const struct hw_module_t* module,
        struct sensors_control_device_t** device) {
    return module->methods->open(module,
            SENSORS_HARDWARE_CONTROL, (struct hw_device_t**)device);
}
```



at hardware/libhardware/include/hardware/sensors.h

◆最後工作：實作 HAL Stub

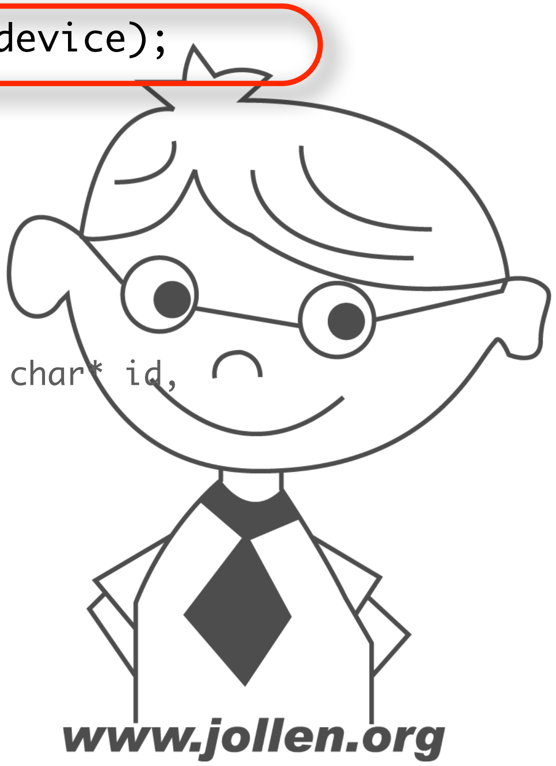
moko365.com



◆HAL Stub 在 open callback function 裡再提供 struct hw_device_t (device control ops)

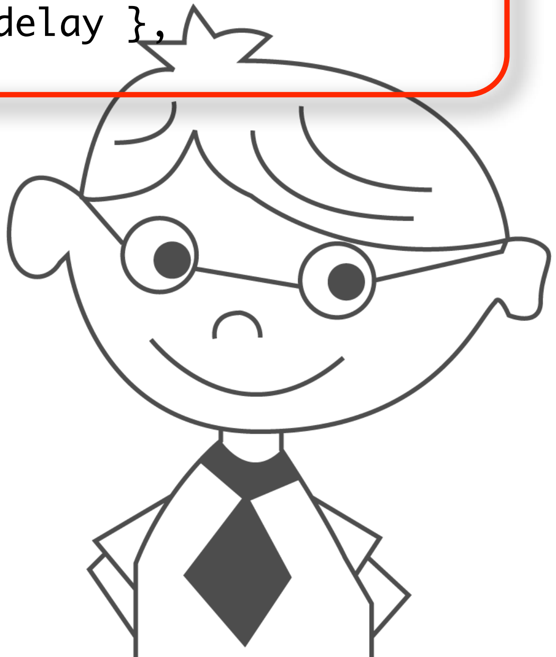
```
static inline int sensors_control_open(const struct hw_module_t* module,
        struct sensors_control_device_t** device) {
    return module->methods->open(module,
            SENSORS_HARDWARE_CONTROL, (struct hw_device_t**)device);
}
```

```
struct hw_module_methods_t {
    /** Open a specific device */
    int (*open)(const struct hw_module_t* module, const char* id,
            struct hw_device_t** device);
};
```



at hardware/libhardware/include/hardware/sensors.h moko365.com

```
static JNINativeMethod gMethods[] = {
    {"_sensors_control_init",     "()I",   (void*) android_init },
    {"_sensors_control_open",     "()Landroid/os/ParcelFileDescriptor;",
                                           (void*) android_open },
    {"_sensors_control_activate", "(IZ)Z", (void*) android_activate },
    {"_sensors_control_wake",     "()I", (void*) android_data_wake },
    {"_sensors_control_set_delay","(I)I", (void*) android_set_delay },
};
```

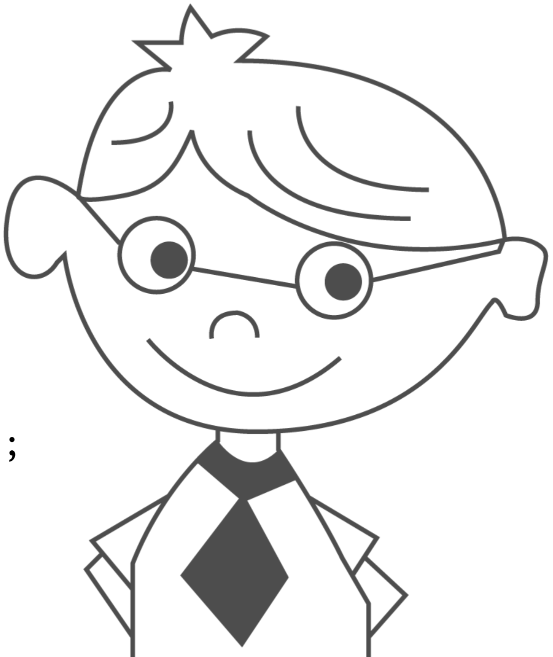


www.jollen.org

◆Device Data Structure

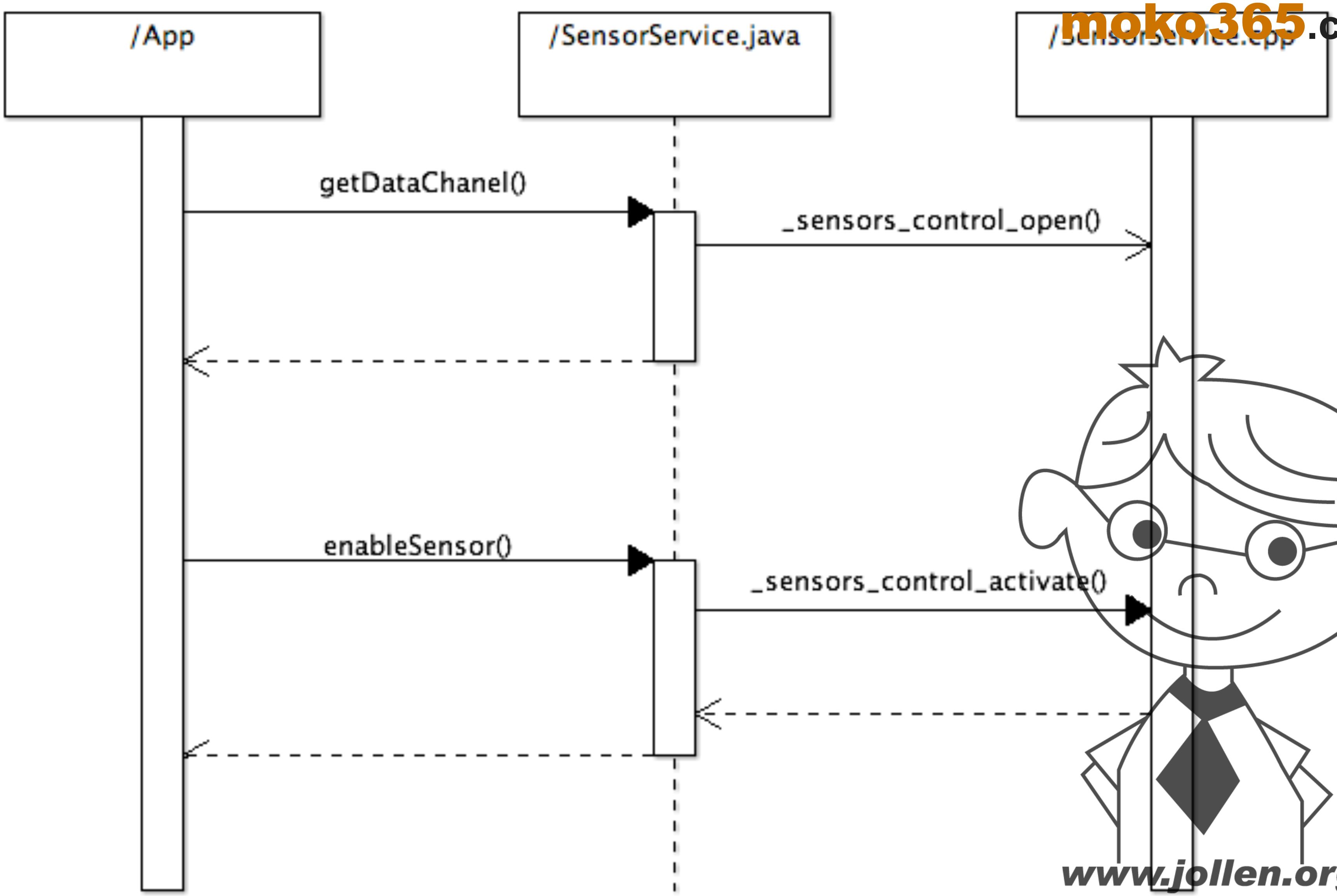
moko365.com

```
/**
 * Every device data structure must begin with hw_device_t
 * followed by module specific public methods and attributes.
 */
struct sensors_control_device_t {
    struct hw_device_t common;

    /**
     * Returns the fd which will be the parameter to
     * sensors_data_device_t::open_data().
     * The caller takes ownership of this fd. This is intended to be
     * passed cross processes.
     *
     * @return a fd if successful, < 0 on error
     */
    int (*open_data_source)(struct sensors_control_device_t *dev);

    int (*activate)(struct sensors_control_device_t *dev,
            int handle, int enabled);

    int (*set_delay)(struct sensors_control_device_t *dev, int32_t ms);

    int (*wake)(struct sensors_control_device_t *dev);
};
```



www.jollen.org

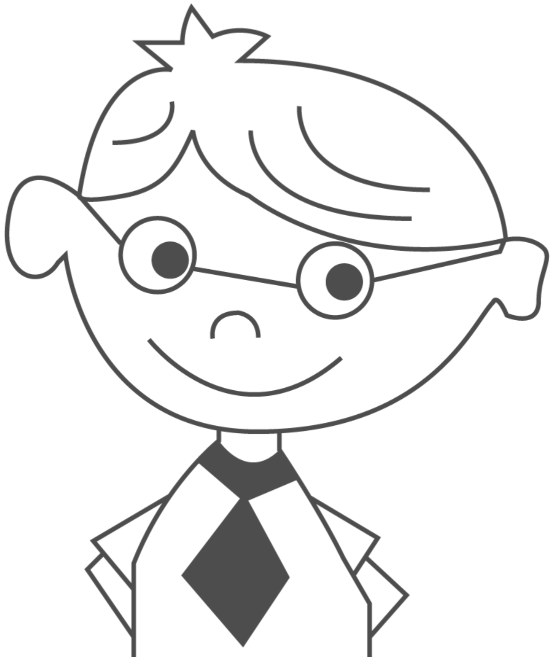
◆Supporting API 其他例子



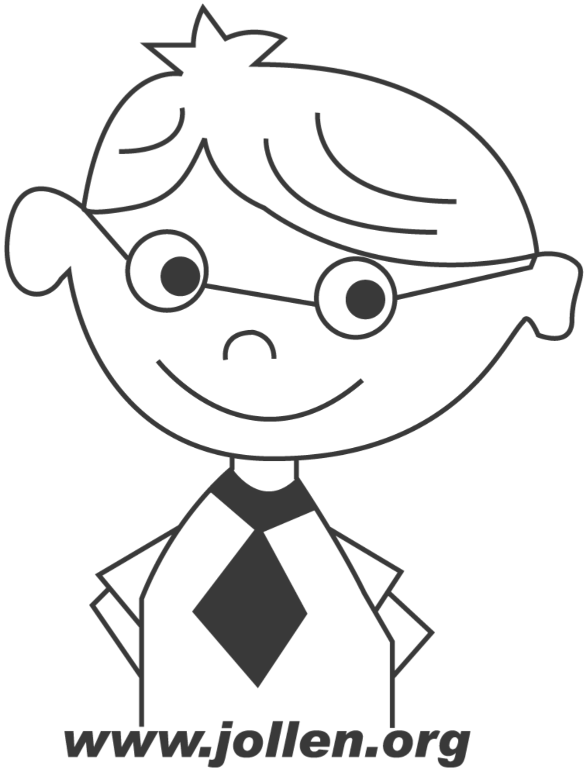
◆hw_device_t

moko365.com

```
/**
 * Every device data structure must begin with hw_device_t
 * followed by module specific public methods and attributes.
 */
struct hw_device_t {
    /** tag must be initialized to HARDWARE_DEVICE_TAG */
    uint32_t tag;

    /** version number for hw_device_t */
    uint32_t version;

    /** reference to the module this device belongs to */
    struct hw_module_t* module;

    /** padding reserved for future use */
    uint32_t reserved[12];

    /** Close this device */
    int (*close)(struct hw_device_t* device);
};
```



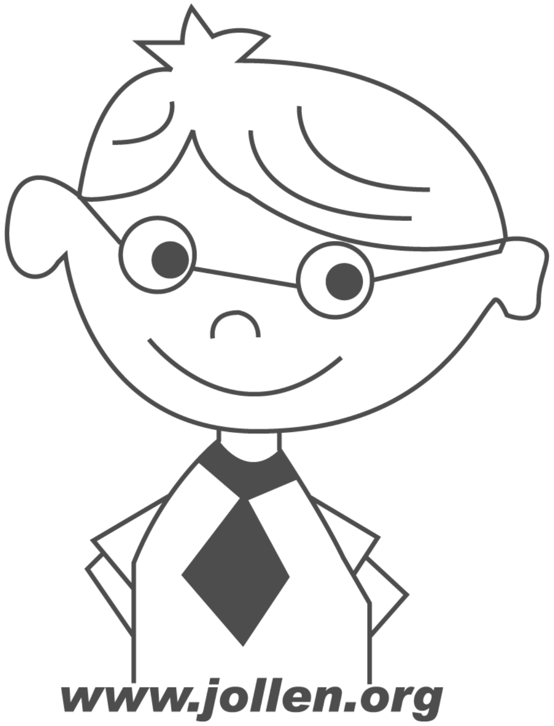
www.jollen.org



Talk V

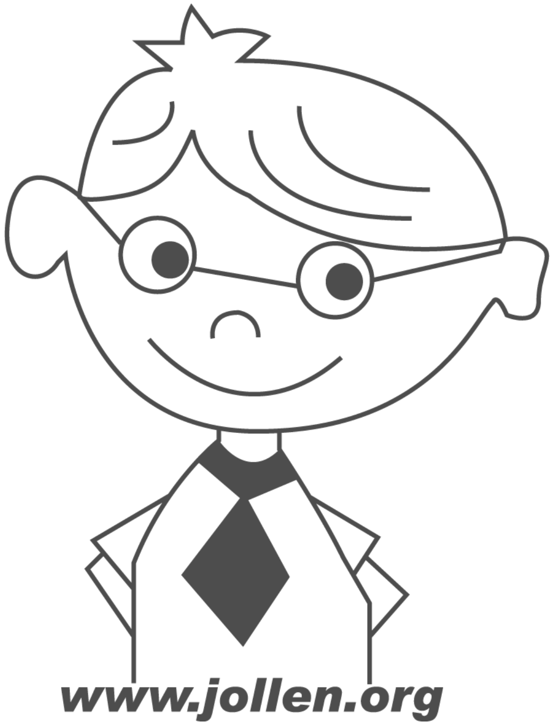
HAL Module 檔案命名

- <MODULE_ID>.variant.so
- led.default.so
- led.smdk6410.so
- /system/lib/hw



HAL Property

- HAL在載入HAL Module前、會試圖取得property
- 透過variant key定義property



加入 Hardware Module

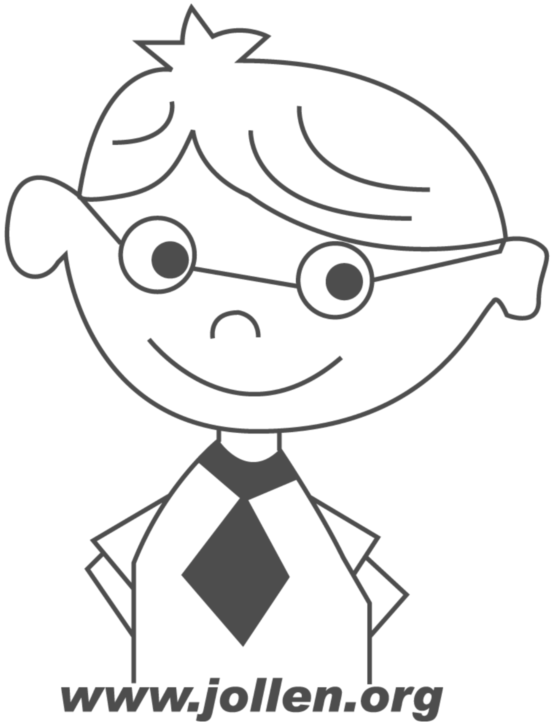
moko365.com

```
on boot
    setprop ro.product.board smdk6410
```

varient key prop

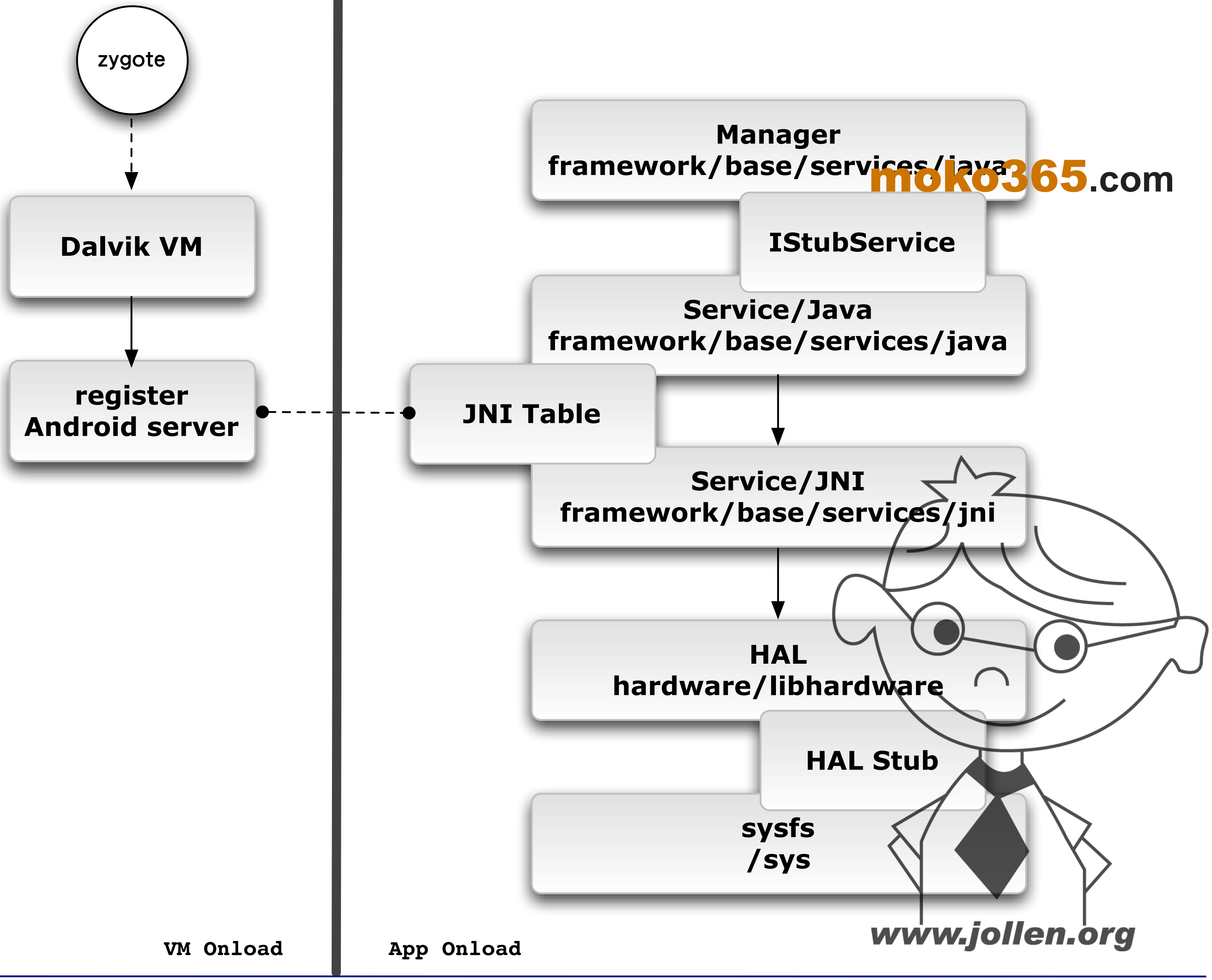
```
/system/lib/hw/sensors.smdk6410.so
```

<MODULE_ID>.<prop>.so



格式：

```
setprop <variant_key> <property>
```



zygote

Dalvik VM

register
Android server

Manager
framework/base/services/java

IStubService

Service/Java
framework/base/services/java

JNI Table

Service/JNI
framework/base/services/jni

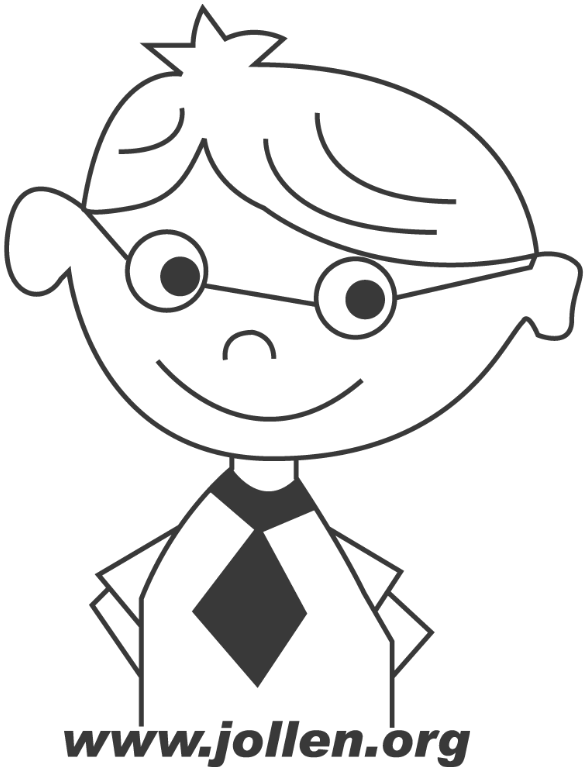
HAL
hardware/libhardware

HAL Stub

sysfs
/sys

VM Onload

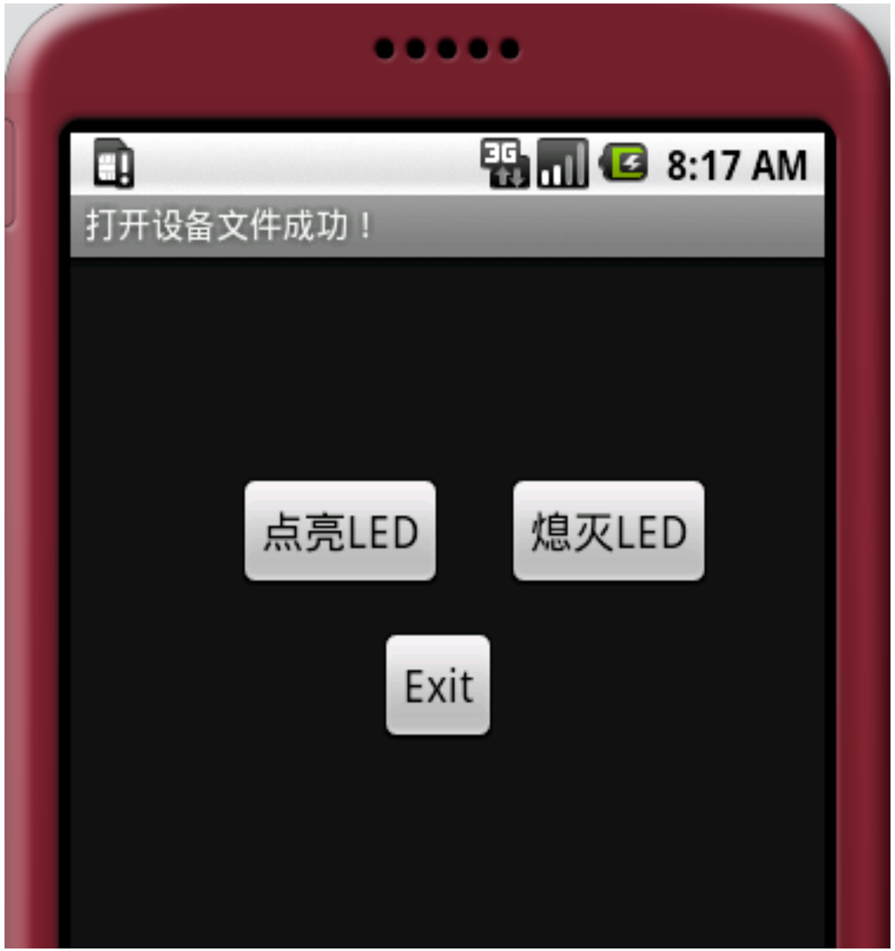
App Onload



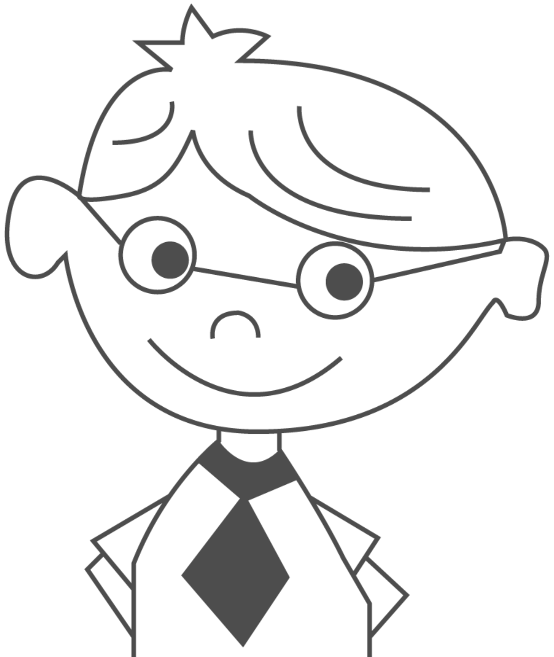
實例補充：不使用 Service 的做法

撰寫 LED 控制應用

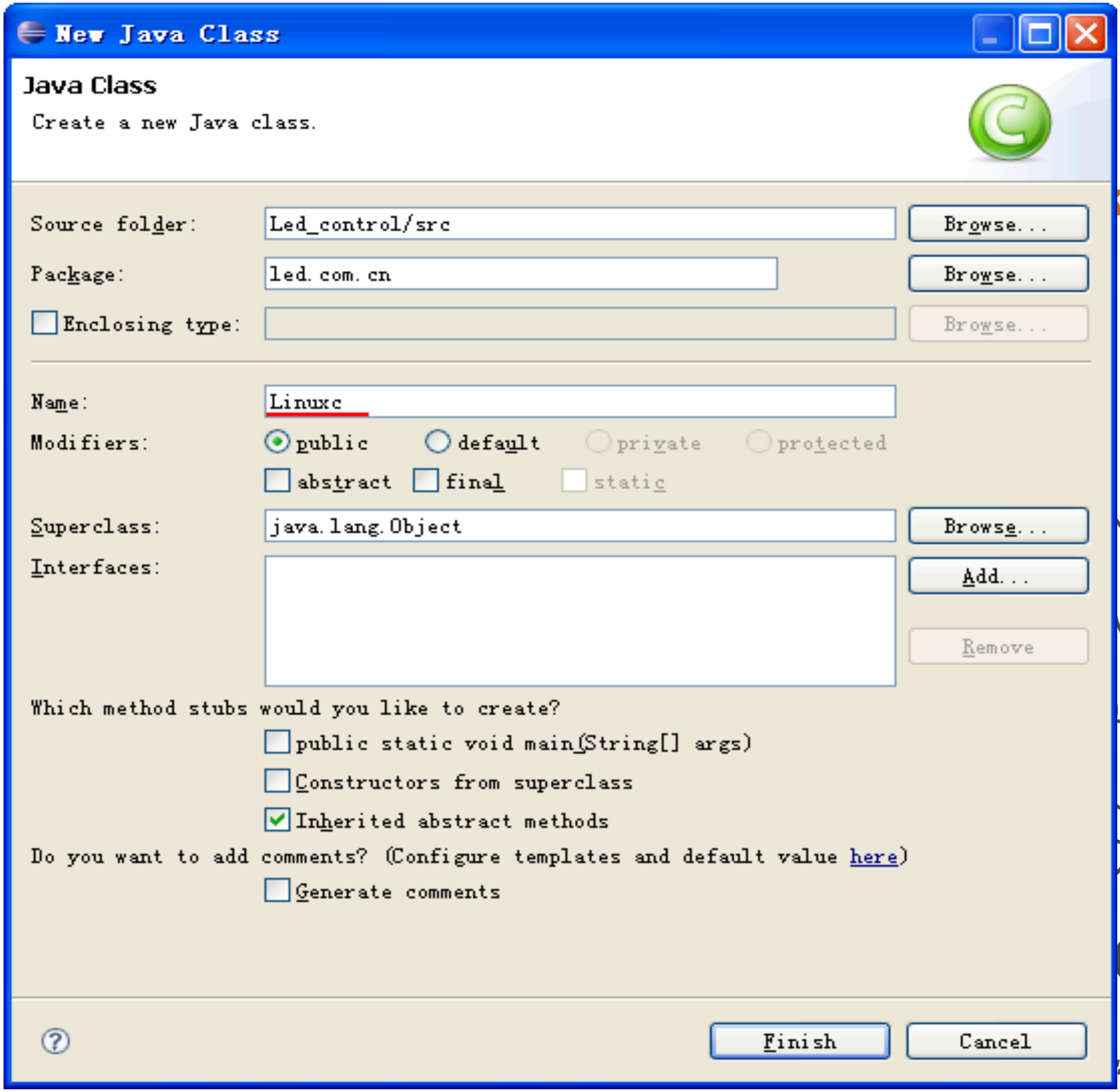
moko365.com



打开设备文件成功！



www.jollen.org



New Java Class

Java Class

Create a new Java class.

Source folder: Led_control/src

Package: led.com.cn

□ Enclosing type:

Name: Linuxc

Modifiers: ● public ○ default ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

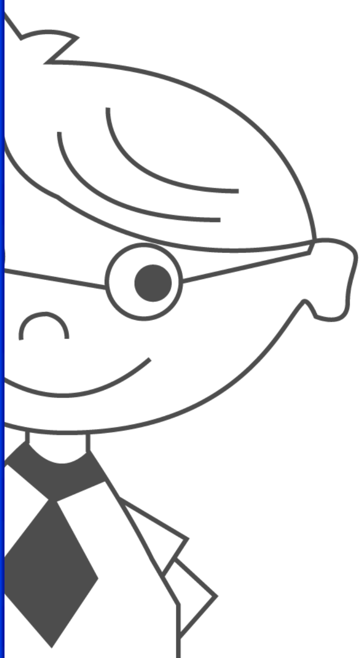
□ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

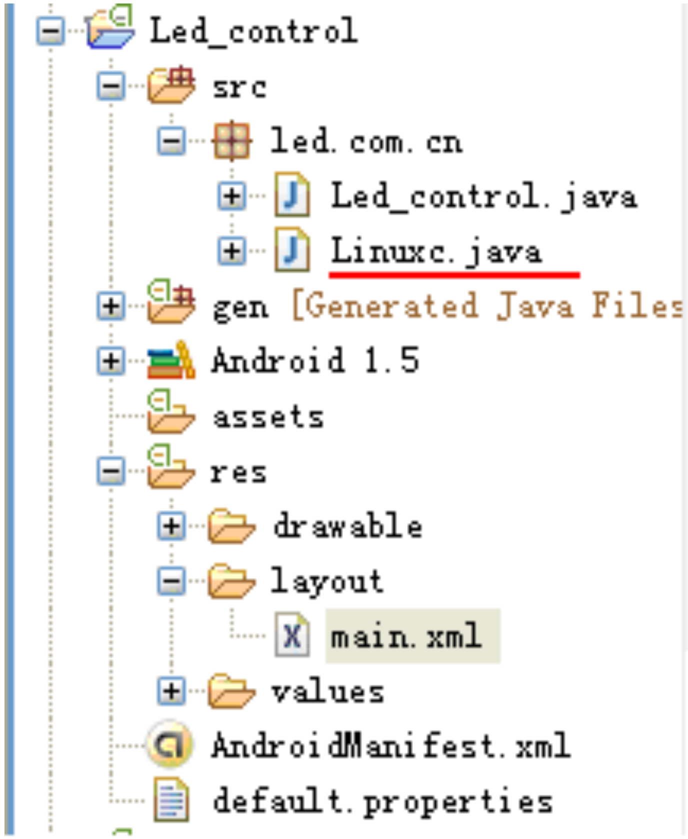
Do you want to add comments? (Configure templates and default value here)

□ Generate comments

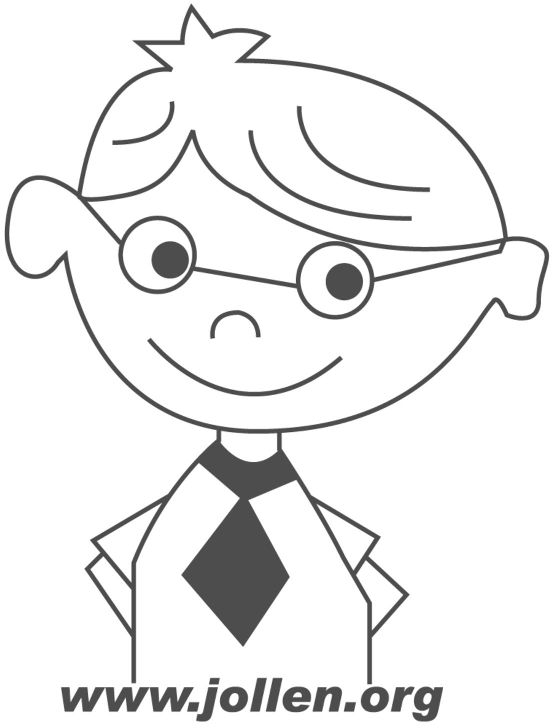


◆Linuxc.java

moko365.com



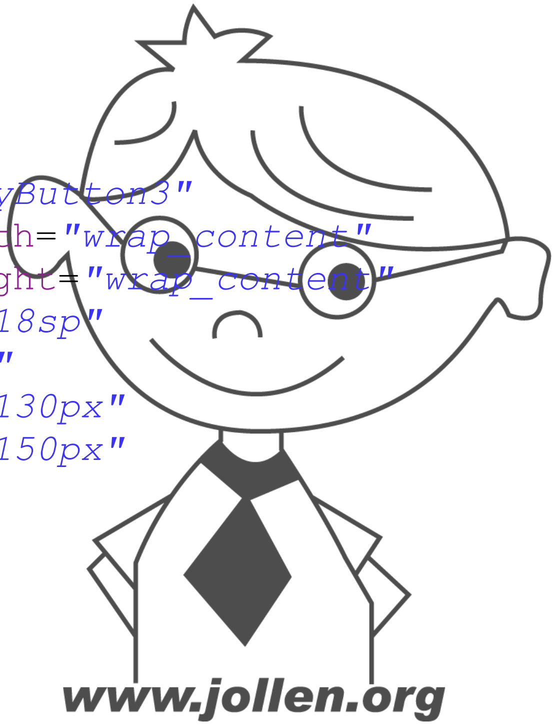
```
Led_control
    src
        led.com.cn
            Led_control.java
            Linuxc.java
    gen [Generated Java Files
    Android 1.5
    assets
    res
        drawable
        layout
            main.xml
        values
    AndroidManifest.xml
    default.properties
```



```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<Button
android:id="@+id/myButton1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="18sp"
android:text="点亮LED"
android:layout_x="70px"
android:layout_y="88px"
>
</Button>
<Button
android:id="@+id/myButton2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="熄灭LED"
android:textSize="18sp"
android:layout_x="184px"
android:layout_y="88px"
>
</Button>
```

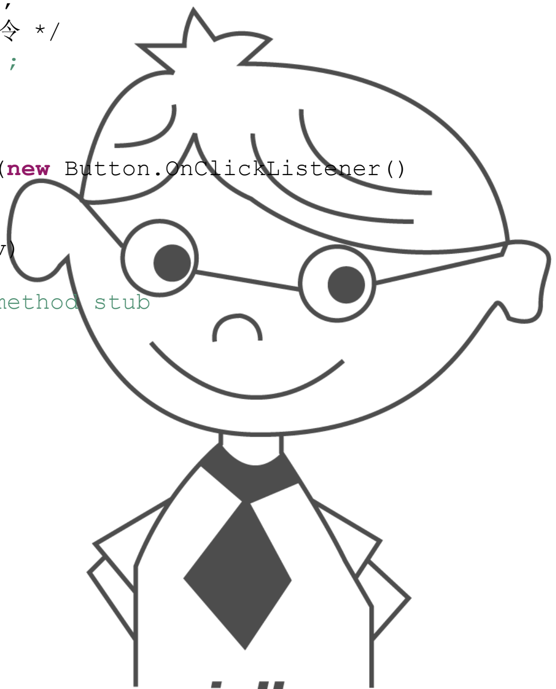
moko365.com

```
<Button
android:id="@+id/myButton3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="18sp"
android:text="Exit"
android:layout_x="130px"
android:layout_y="150px"
>
</Button>
</AbsoluteLayout>
```



```
package led.com.cn;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class Led_control extends Activity {
    /** Called when the activity is first created. */
      /* 定义3个Button，声明为private类型 */
     private Button mButton1;
      private Button mButton2;
      private Button mButton3;

      /* 定义要控制LED的编号 */
     public  int num = 4;
      /* 1为点亮 */
     public  int led_on  = 1;
      /* 2为熄灭 */
      public  int led_off = 2;
      public  int fd = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mButton1 =(Button) findViewById(R.id.myButton1);
        mButton2 =(Button) findViewById(R.id.myButton2);
        mButton3 =(Button) findViewById(R.id.myButton3);

        /*  打开led设备文件，并得到一个返回值fd */
        fd = Linuxc.openled();
        if (fd < 0){
            setTitle("设备文件不存在！");
            finish();
         /* 打开设备文件失败的话，就退出 */
        }
        else {
            setTitle("打开设备文件成功！");
        }}
```

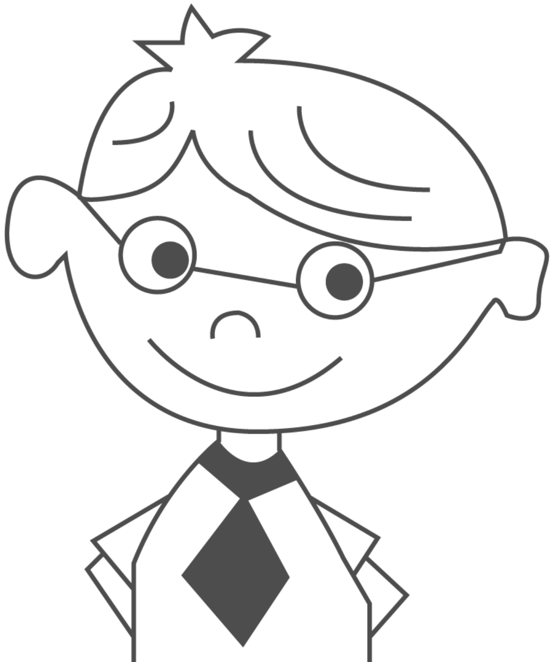
```
 /*使用setOnClickListener来监听事件*/
mButton1.setOnClickListener(new Button.OnClickListener()
{
  @Override
 /* 使用onClick 来响应事件 */
  public void onClick(View v)
  {
    // TODO Auto-generated method stub
       setTitle("LED点亮了！");
   /* 给编号为4的LED发送点亮的指令 */
    Linuxc.send(num, led_on);

  }
});

mButton2.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    // TODO Auto-generated method stub
      setTitle("LED熄灭了！");
  /* 给编号为4的LED发送熄灭的指令 */
   Linuxc.send(num, led_off);
    }
});

mButton3.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    // TODO Auto-generated method stub
   /* 关闭设备文件 */
      Linuxc.closeled();
  /*  退出运用程序 */
      finish();
    }
});

                          }
                       }
```



www.jollen.org

```
package led.com.cn;

import android.util.Log;

public class Linuxc {
    static {
      try {
         Log.i("JNI", "Trying to load libled.so");
            /* 加载libled.so 库 */
         System.loadLibrary("led");
      }
      catch (UnsatisfiedLinkError ule) {
         Log.e("JNI", "WARNING: Could not load libled.so");
      }}
    /* 声明openled()为本地方法  */
    public static native int openled();
    /* 声明closeled ()为本地方法  */
    public static native int closeled();
    /* 声明send()为本地方法  */
    public static native int send(int led_num, int on_off);

}
```

moko365.com

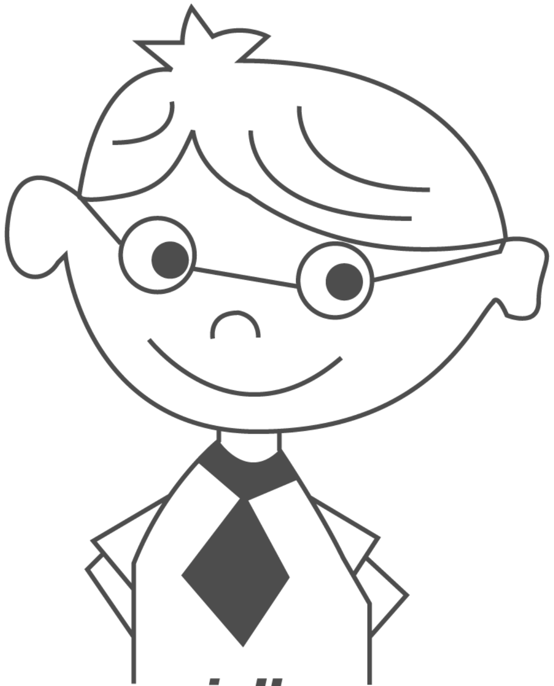


www.jollen.org

led_com_cn_Linuxc.h

- 将工程文件夹Led_control拷贝到ubuntu的/home/online目录下，新建一个文件夹led_test
- 用JDK产生与Linuxc.class相应的头文件，用做JNI接口函数声明
- 产生，led_com_cn_Linuxc.h 头文件

```
$ mkdir led_test
$ cd led_test
...
$ /javah -classpath ~/Led_Control/bin/ led.com.cn.Linuxc
```

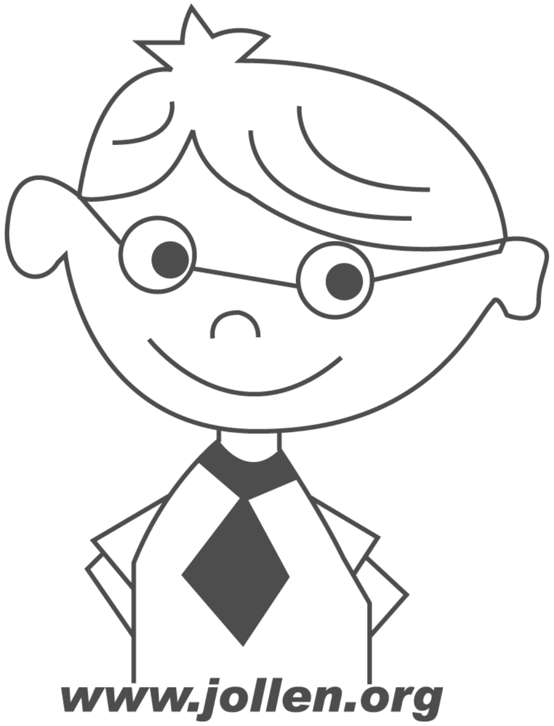


www.jollen.org

◆led_com_cn_Linuxc.c

moko365.com

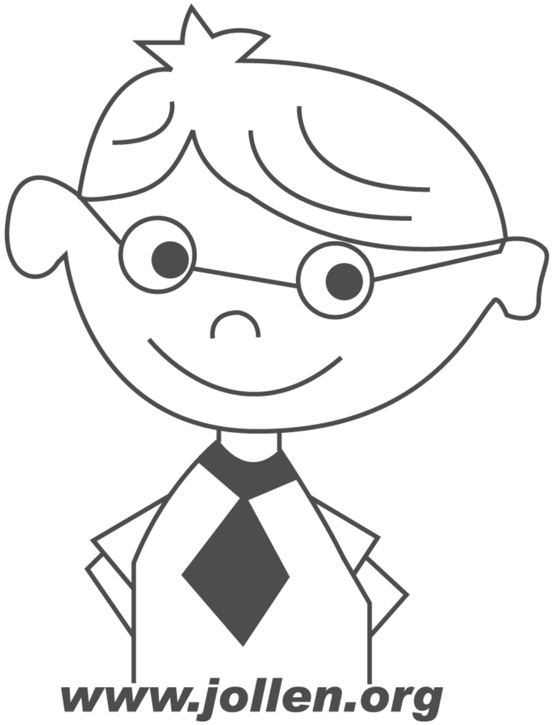
```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/ioctl.h>

#include "led_com_cn_Linuxc.h"
#include "led.h"

#define LED_TEST 3

#define DEVICE_BLTEST "/dev/led"

int fd;

JNIEXPORT jint JNICALL Java_led_com_cn_Linuxc_openled
 (JNIEnv *env, jclass mc)
{
 fd= open(DEVICE_BLTEST,O_RDONLY);
 return fd;
}

JNIEXPORT jint JNICALL Java_led_com_cn_Linuxc_closeled
 (JNIEnv *env, jclass mc)
{
 close(fd);
}

JNIEXPORT jint JNICALL Java_led_com_cn_Linuxc_send
(JNIEnv *env, jclass mc, jint a, jint b)
{
 ioctl(fd,b,&a);
}
```



編譯 libled.so

moko365.com

```
$  arm-none-linux-gnueabi-gcc    -I/home/online/jdk1.6.0_14/include  -I/home/online/jdk1.6.0_14/include/linux -fpic -c led_com_cn_linuxc.c

$arm-none-linux-gnueabi-ld-T    /home/online/CodeSourcery/Sourcery_G++_Lite/arm-none-linux-gnueabi/lib/ldscripts/armelf_linux_eabi.xsc -shared -o libled.so led_com_cn_linuxc.o
```



www.moko365.com