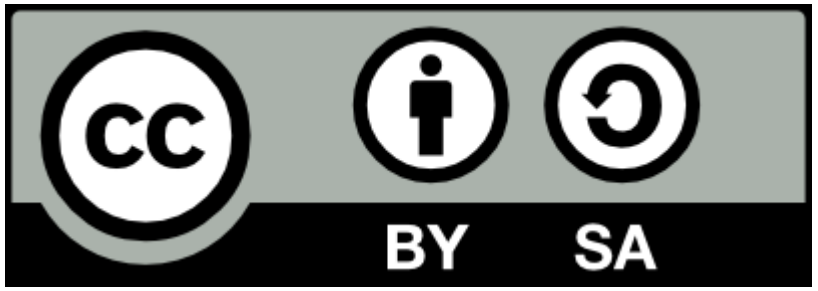
Running Code in the Android Stack

Embedded Linux Conference Europe 2013

Karim Yaghmour
@karimyaghmour

karim.yaghmour@opersys.com



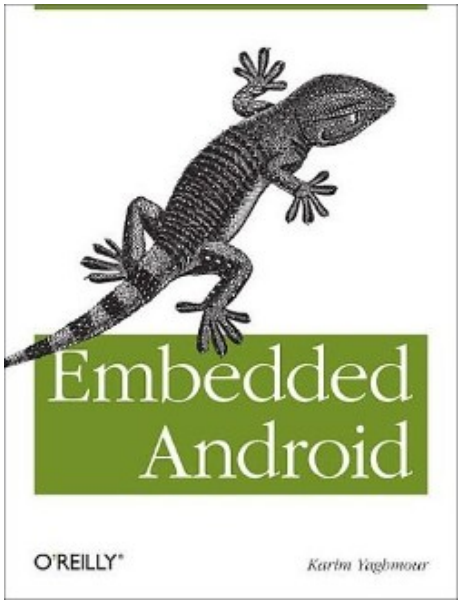
These slides created by: Karim Yaghmour

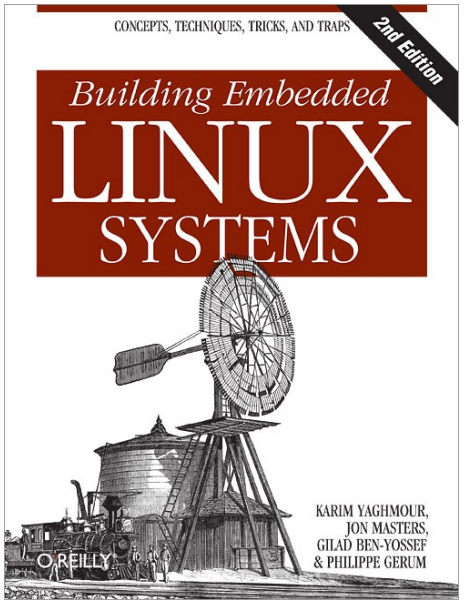
Originals at: www.opersys.com/community/docs

Delivered and/or customized by

About

- Author of:





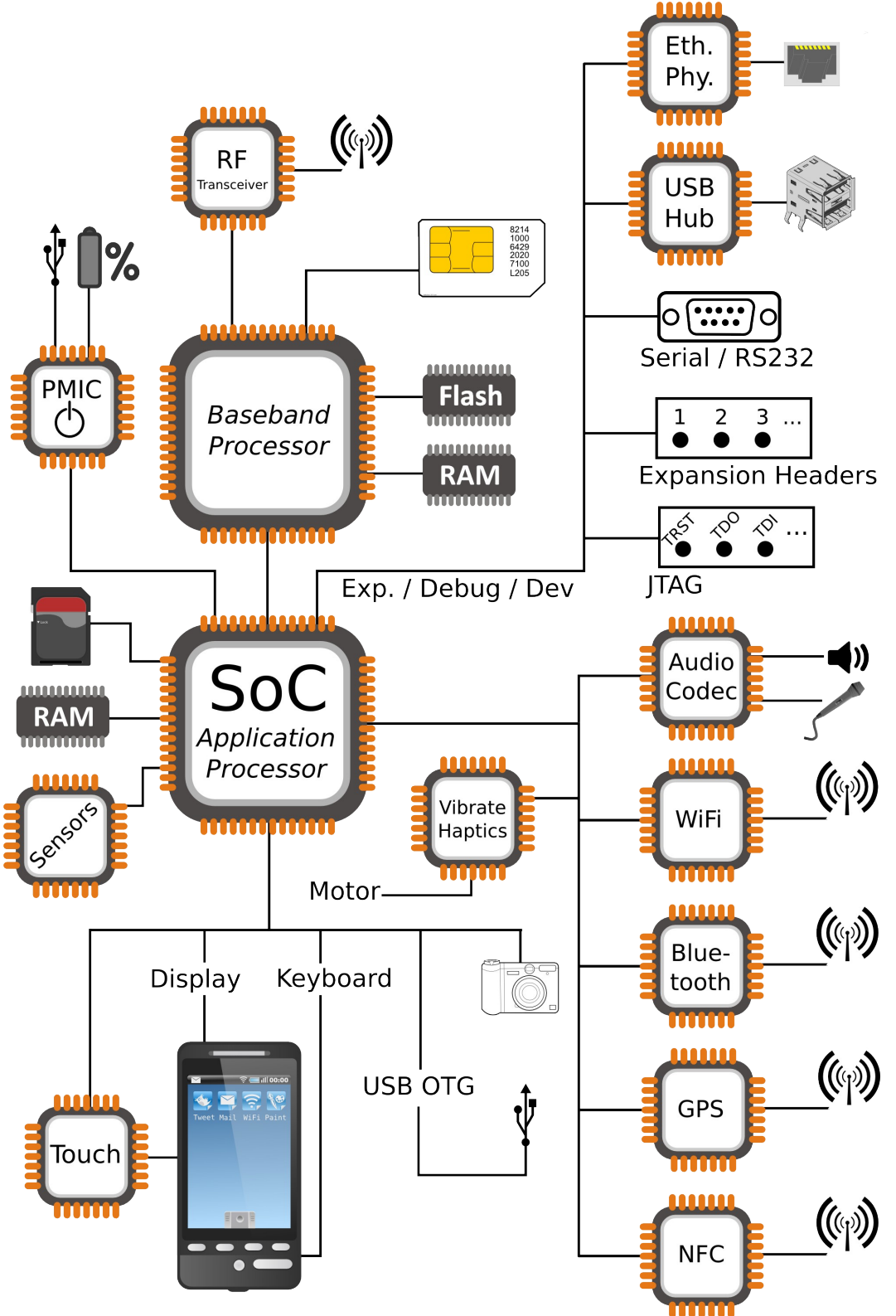
- Introduced Linux Trace Toolkit in 1999
- Originated Adeos and relayfs (kernel/relay.c)
- Training, Custom Dev, Consulting, ...

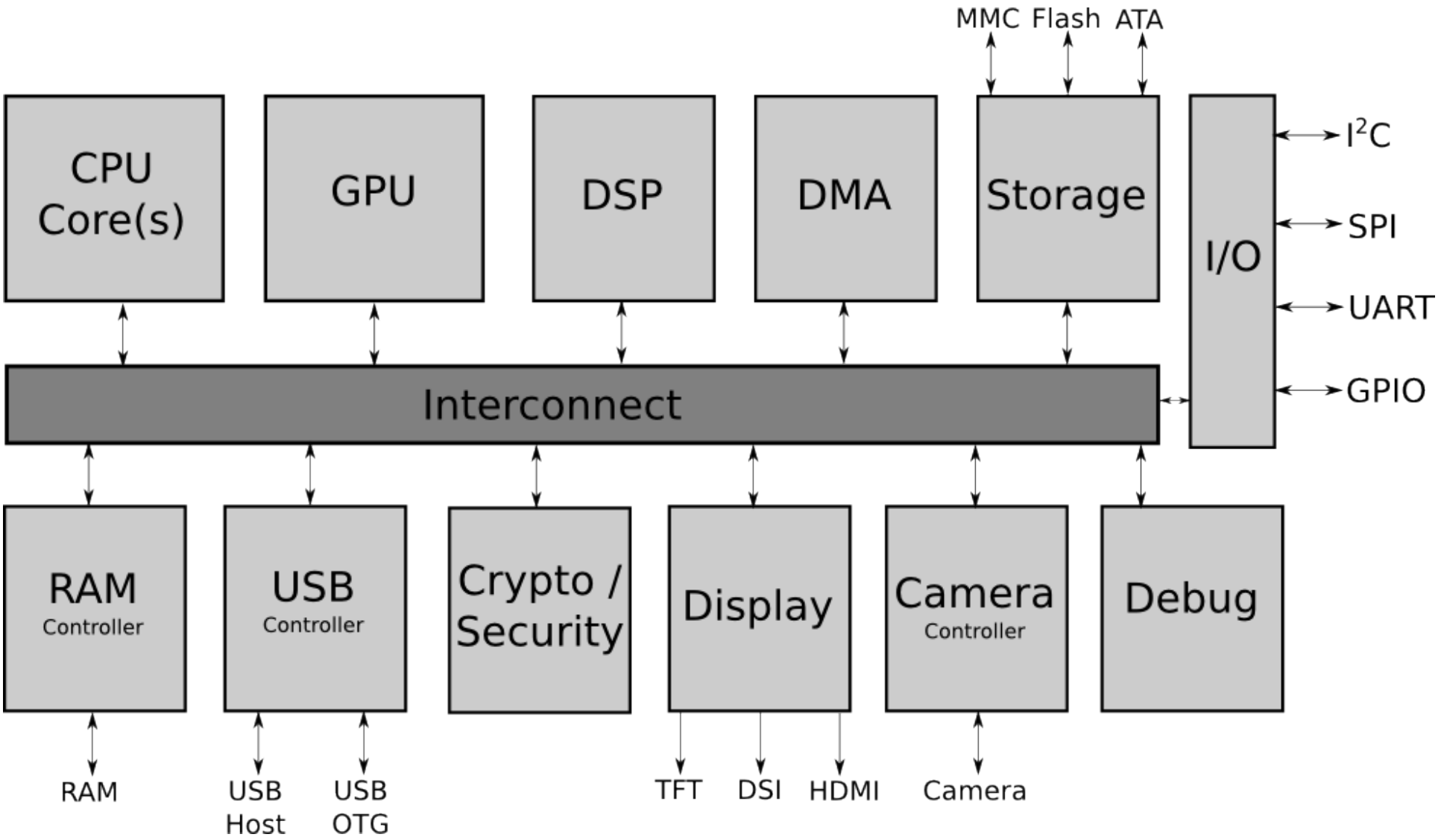
Agenda

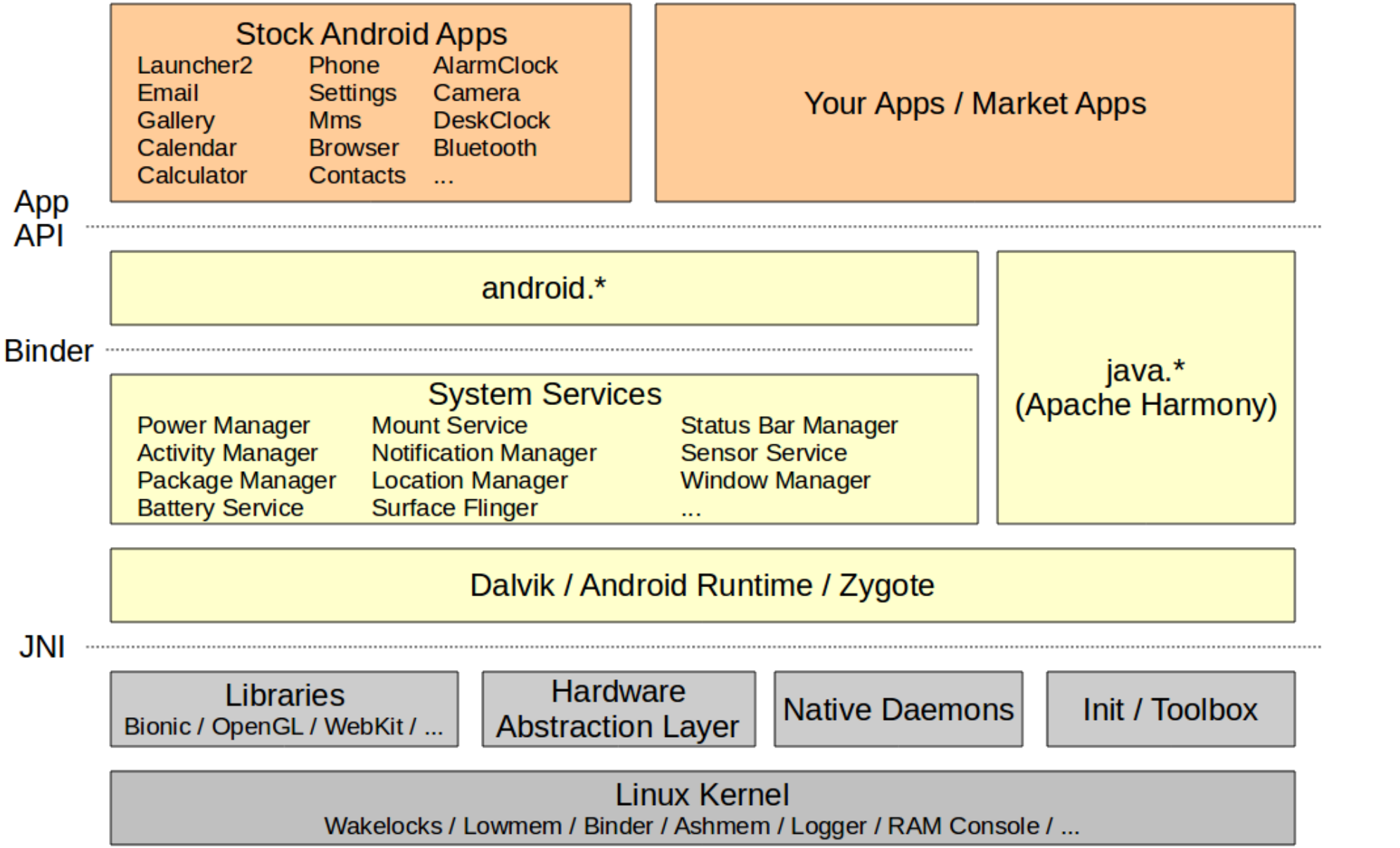
1. Architecture Basics
2. Programming Languages
3. Standard App Mechanisms
4. Special App Mechanisms
5. Starting Apps
6. Native Utilities and Daemons
7. Java Utilities (and Daemons)
8. System Services
9. Shell Scripts
10. init.rc Commands and Services
11. C Libraries
12. Java Libraries
13. SDK add-ons
14. Legacy Linux

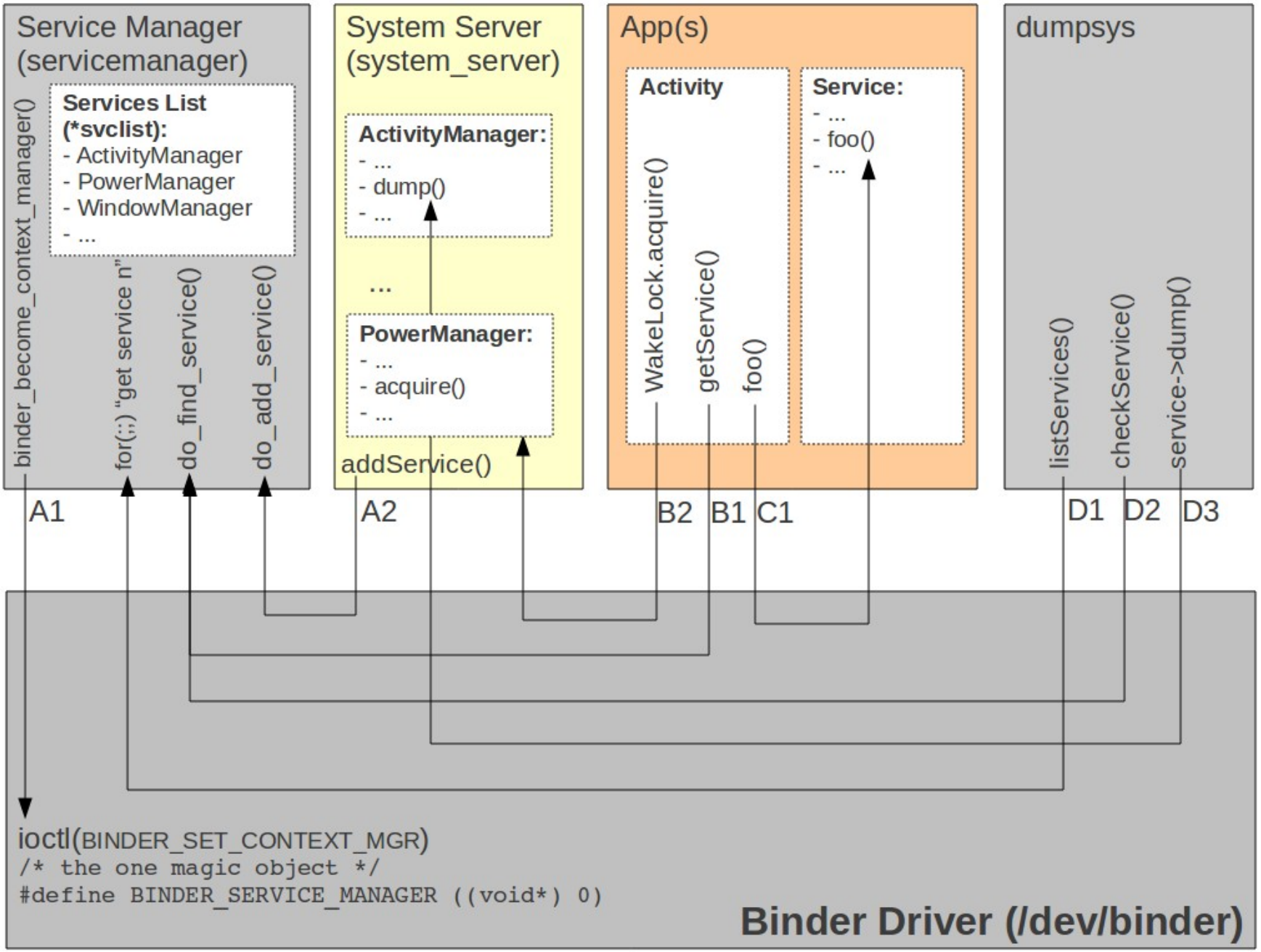
1. Architecture Basics

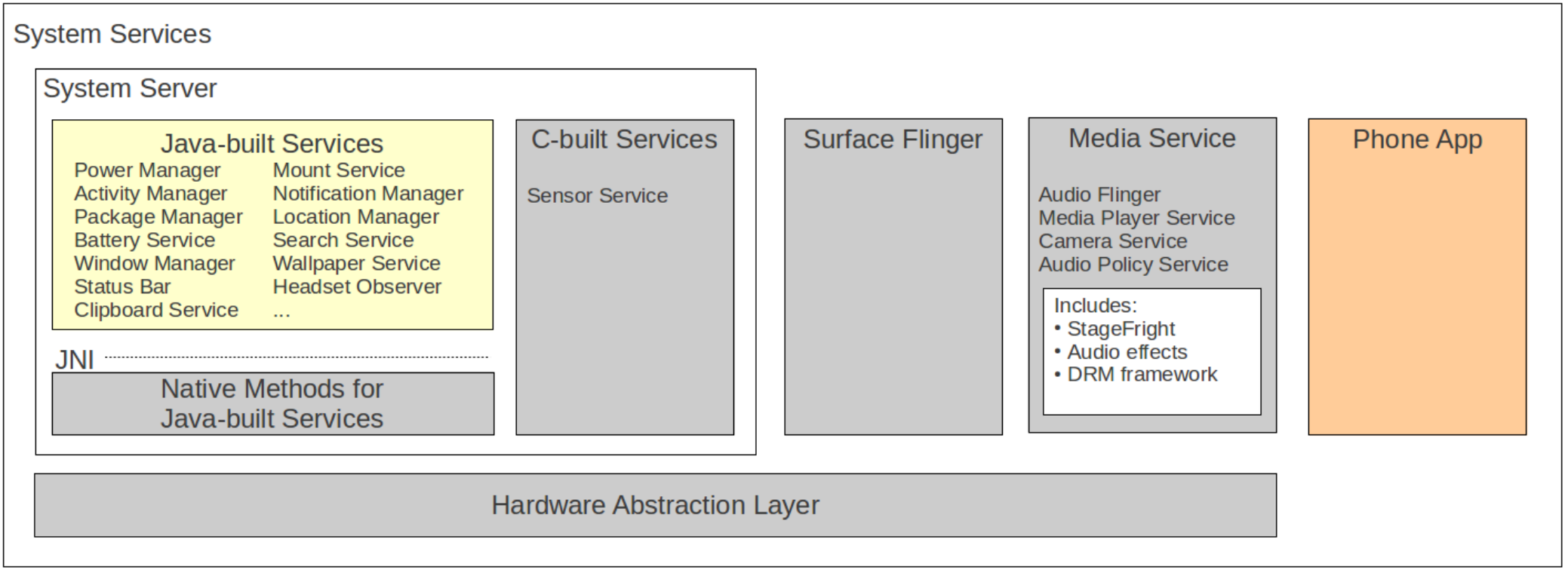
- Hardware used to run Android
- AOSP
- Binder
- System Services
- HAL

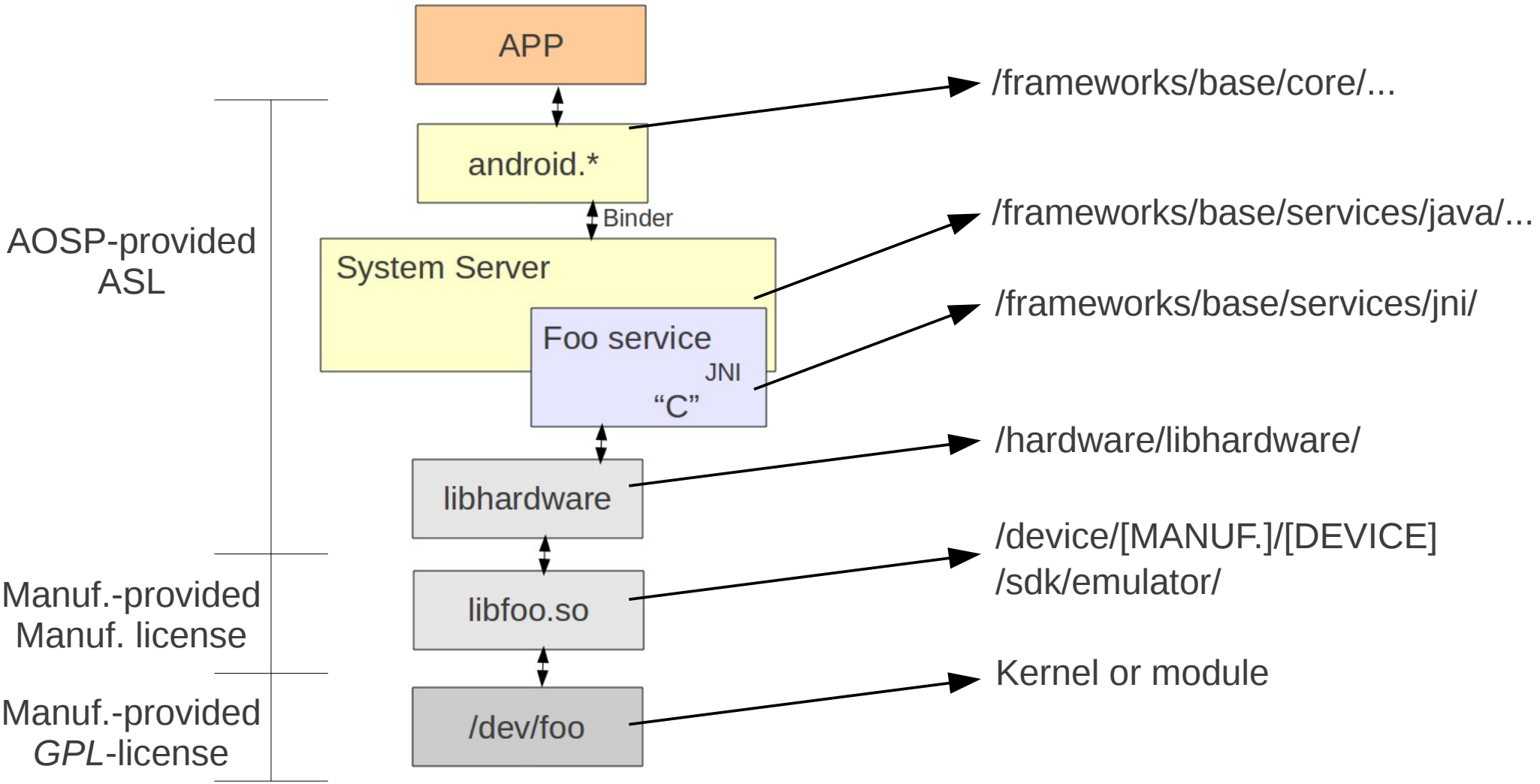

Eth.
Phy.
RF
Transceiver
USB
Hub
8214
1000
6429
2020
7100
L205
Serial / RS232
PMIC
Baseband
Processor
Flash
RAM
1 2 3 ...
Expansion Headers
TRST TDO TDI ...
JTAG
Exp. / Debug / Dev
SoC
Application
Processor
RAM
Audio
Codec
Sensors
Vibrate
Haptics
WiFi
Motor
Blue-
tooth
Display
Keyboard
GPS
Touch
USB OTG
NFC


MMC Flash ATA
CPU Core(s)
GPU
DSP
DMA
Storage
I/O
I²C
SPI
UART
GPIO
Interconnect
RAM Controller
USB Controller
Crypto / Security
Display
Camera Controller
Debug
RAM
USB Host
USB OTG
TFT
DSI
HDMI
Camera


Stock Android Apps
Launcher2 Phone AlarmClock
Email Settings Camera
Gallery Mms DeskClock
Calendar Browser Bluetooth
Calculator Contacts ...
Your Apps / Market Apps
App API
android.*
Binder
System Services
Power Manager Mount Service Status Bar Manager
Activity Manager Notification Manager Sensor Service
Package Manager Location Manager Window Manager
Battery Service Surface Flinger ...
java.*
(Apache Harmony)
Dalvik / Android Runtime / Zygote
JNI
Libraries
Bionic / OpenGL / WebKit / ...
Hardware Abstraction Layer
Native Daemons
Init / Toolbox
Linux Kernel
Wakelocks / Lowmem / Binder / Ashmem / Logger / RAM Console / ...


Service Manager
(servicemanager)
Services List
(*svclist):
- ActivityManager
- PowerManager
- WindowManager
- ...
binder_become_context_manager()
for(;;) "get service n"
do_find_service()
do_add_service()
System Server
(system_server)
ActivityManager:
- ...
- dump()
- ...
...
PowerManager:
- ...
- acquire()
- ...
addService()
App(s)
Activity
WakeLock.acquire()
getService()
foo()
Service:
- ...
- foo()
- ...
dumpsys
listServices()
checkService()
service->dump()
A1
A2
B2
B1
C1
D1
D2
D3
ioctl(BINDER_SET_CONTEXT_MGR)
/* the one magic object */
#define BINDER_SERVICE_MANAGER ((void*) 0)
Binder Driver (/dev/binder)


System Services
System Server
Java-built Services
Power Manager
Activity Manager
Package Manager
Battery Service
Window Manager
Status Bar
Clipboard Service
Mount Service
Notification Manager
Location Manager
Search Service
Wallpaper Service
Headset Observer
...
JNI
Native Methods for Java-built Services
C-built Services
Sensor Service
Surface Flinger
Media Service
Audio Flinger
Media Player Service
Camera Service
Audio Policy Service
Includes:
• StageFright
• Audio effects
• DRM framework
Phone App
Hardware Abstraction Layer


APP
android.*
Binder
System Server
Foo service
JNI
"C"
libhardware
libfoo.so
/dev/foo
/frameworks/base/core/...
/frameworks/base/services/java/...
/frameworks/base/services/jni/
/hardware/libhardware/
/device/[MANUF.]/[DEVICE]
/sdk/emulator/
Kernel or module
AOSP-provided
ASL
Manuf.-provided
Manuf. license
Manuf.-provided
GPL-license

2. Programming Languages

- Java:
  - Apps
  - Platform
- C:
  - Apps
  - Platform
- JavaScript / CSS / HTML
  - WebKit object
- C#:
  - Mono for Android
- Misc.
  - Any language for which there's a Linux compiler / interpreter

3. Standard App Mechanisms

- Components:
  - Activity
  - Service
  - Content Provider
  - Broadcast Receiver
- “Application” Component
- Widgets

3.1. Components

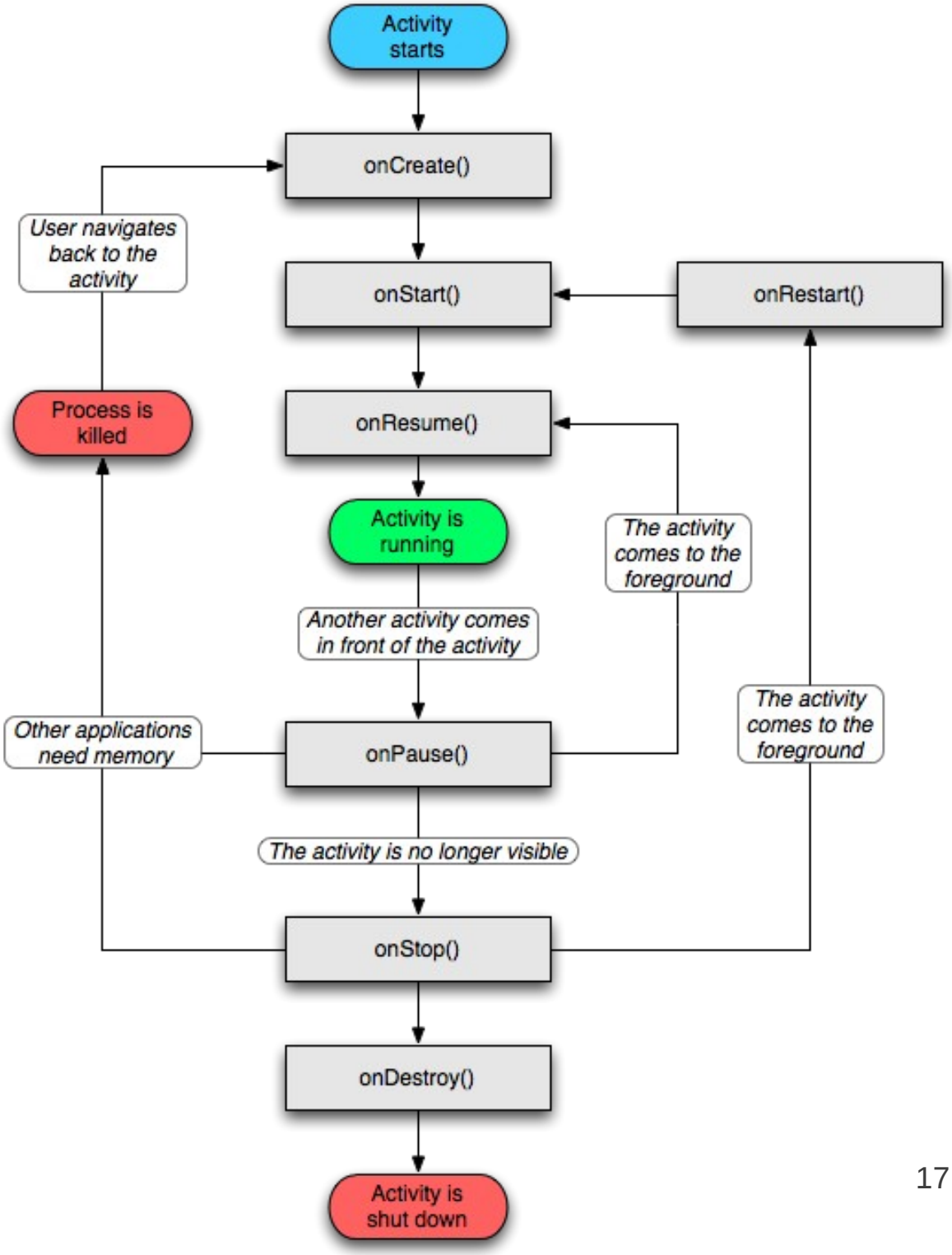
- 1 App = N Components
- Apps can use components of other applications
- App processes are automagically started whenever any part is needed
- Ergo: N entry points, !1, and !main()
- Components:
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers

3.2. Intents

- Intent = asynchronous message w/ or w/o designated target
- Like a polymorphic Unix signal, but w/o required target
- Intents “payload” held in Intent Object
- Intent Filters specified in Manifest file

3.3. Component lifecycle

- System automagically starts/stops/kills processes:
  - Entire system behaviour predicated on low memory
- System triggers Lifecycle callbacks when relevant
- Ergo: Must manage Component Lifecycle
- Some Components are more complex to manage than others


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

3.4. Manifest file

- Informs system about app's components
- XML format
- Always called AndroidManifest.xml
- Components
  - Activity = <activity> ... static
  - Service = <service> ... static
  - Broadcast Receiver:
    - Static = <receiver>
    - Dynamic = Context.registerReceiver()
  - Content Provider = <provider> ... static
- Intent filters

3.5. Processes and threads

- Processes
  - Default: all callbacks to any app Component are issued to the main process thread
  - <activity>—<service>—<recipient>—<provider> have process attribute to override default
  - Do NOT perform blocking/long operations in main process thread:
    - Spawn threads instead
  - Process termination/restart is at system's discretion
  - Therefore:
    - Must manage Component Lifecycle
- Threads:
  - Create using the regular Java Thread Object
  - Android API provides thread helper classes:
    - Looper: for running a message loop with a thread
    - Handler: for processing messages
    - HandlerThread: for setting up a thread with a message loop

3.6. Remote procedure calls

- Android RPCs = Binder mechanism
- No Sys V IPC due to in-kernel resource leakage
- Binder is a low-level functionality, not used as-is
- Instead: must define interface using Interface Definition Language (IDL)
- IDL fed to aidl Tool to generate Java interface definitions

3.7. “Application” Component

- See packages/apps/Phone:
  - Especially src/com/android/phone/PhoneApp.java

```
/**
 * Top-level Application class for the Phone app.
 */
public class PhoneApp extends Application {
    PhoneGlobals mPhoneGlobals;

    public PhoneApp() {
    }

    @Override
    public void onCreate() {
        if (UserHandle.myUserId() == 0) {
            // We are running as the primary user, so should bring up the
            // global phone state.
            mPhoneGlobals = new PhoneGlobals(this);
            mPhoneGlobals.onCreate();
        }
    }

...
```

3.8. Widgets

- See:

  https://developer.android.com/guide/topics/appwidgets

4. Special App Mechanisms

- Foreground services
- Persistent apps
- Sync adapters
- Backup agents
- Input methods engines
- Alarm services
- Live wallpapers
- Account managers
- Device administrators
- “Core” app

4.1. Foreground services

- Sticky notification icon
- Ex:
  - Skype
  - Avast
- See:
  - https://developer.android.com/guide/components/services.html#Foreground
  - https://developer.android.com/reference/android/app/Service.html#startForeground%28int,%20android.app.Notification%29
  - https://developer.android.com/reference/android/app/Notification.html#FLAG_FOREGROUND_SERVICE

4.2. Persistent apps

- Flag in <application> decl. in manifest:

  android:persistent="true"
- For “system” apps only
- Will cause app to be kept alive by ActivityManager
- IOW:
  - It'll be automagically restarted if it dies
  - Lifecycle won't result in it dying

4.3. Sync adapters

- For sync'ing with a cloud service
- Typically for REST-based apps
- Ex.: a Twitter-like feed
- See:
  - http://www.google.com/events/io/2010/sessions/developing-RESTful-android-apps.html

4.4. Backup Agents

- Triggered by Backup Manager
- See:
  - https://developer.android.com/guide/topics/data/backup.html

4.5. Input methods Engines

- Virtual on-screen keyboards
- See:
  - https://developer.android.com/guide/topics/text/creating-input-method.html

4.6. Alarm services

- Cause Intent to trigger in the future:
- See:
  - https://developer.android.com/reference/android/app/AlarmManager.html

4.7. Live wallpapers

- Animated wallpapers
- See:
  - http://www.vogella.com/articles/AndroidLiveWallpaper/article.html

4.8. Account managers

- For managing accounts in Settings
- See:
  - https://developer.android.com/reference/android/accounts/AccountManager.html

4.9. Device administrators

- Enterprisification feature:
- See:
  - https://developer.android.com/guide/topics/admin/device-admin.html

4.10. “core” app

- Use this tag in <manifest> tag:

  coreApp="true"
- Used by:
  - StatusBar
  - SettingsProvider
  - See frameworks/base/packages/
- See “onlyCore” variable in SystemServer.java
- Causees only core apps to start if set to “true”
- Default hard-coded as “false”
- PackageManagerService.java has detail of how it's used

5. Starting Apps

- Intents
- Content Resolvers

6. Native Utilities and Daemons

- Compile as part of AOSP:

  BUILD_EXECUTABLE
- Build outside AOSP and merge:
  - Link statically (like adbd), or
  - Link dynamically (against Bionic or glibc or ...)
- Run like any regular Linux command-line tool
- Start from:
  - adb shell
  - init.rc
  - shell script
  - Android app (N.B. This will continue running independently)
- Plenty of existing examples: netd, vold, installd, etc.

6.1. Start from Android app

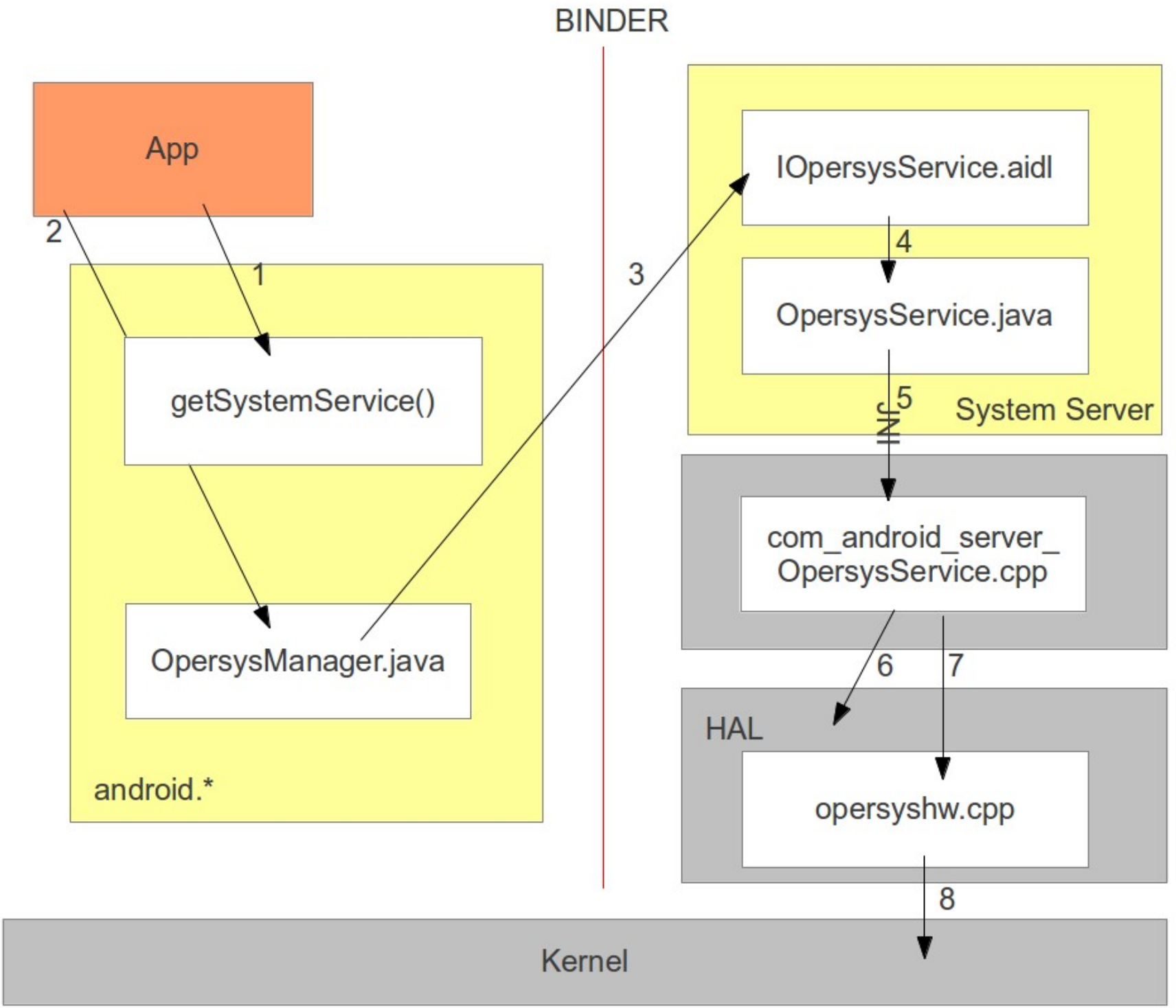
```
Process myUtil;
BufferReader myCliOutput;

myUtil =
    Runtime.getRuntime().exec("my_cli_util -P params");

myCliOutput =
    new BufferedReader(
        new InputStreamReader(
                myUtil.getInputStream()));
```

7. Java Utilities (and Daemons)

- Coded as a “library”
- Launched using app_process utility
- Use script to wrap call to app_process
- See frameworks/base/cmds for examples:
  - am
  - pm
  - svc
  - ...
- No known case of “daemon”, but should be feasible
  - Closest is System Server, but it's started from Zygote, not directly from app_process

8. System Services

- Start in:
  - Existing Java system services
  - App (like Phone or NFC)
  - C daemon (like Media Service or SurfaceFlinger)
- Register with Service Manager
- Provide API speak with system service:
  - In existing framework
  - As SDK add-on


BINDER
App
2
1
getSystemService()
OpersysManager.java
android.*
3
IOpersysService.aidl
4
OpersysService.java
5
JNI
System Server
com_android_server_
OpersysService.cpp
6
7
HAL
opersyshw.cpp
8
Kernel

8.1. OpersysService.java

```
package com.android.server;

import android.content.Context;
import android.os.Handler;
import android.os.IOpersysService;
import android.os.Looper;
import android.os.Message;
import android.os.Process;
import android.util.Log;

public class OpersysService extends IOpersysService.Stub {
    private static final String TAG = "OpersysService";
    private OpersysWorkerThread mWorker;
    private OpersysWorkerHandler mHandler;
    private Context mContext;

    public OpersysService(Context context) {
        super();
        mContext = context;
        mWorker = new OpersysWorkerThread("OpersysServiceWorker");
        mWorker.start();
        Log.i(TAG, "Spawned worker thread");
    }

    public void setValue(int val) {
        Log.i(TAG, "setValue " + val);
        Message msg = Message.obtain();
        msg.what = OpersysWorkerHandler.MESSAGE_SET;
        msg.arg1 = val;
        mHandler.sendMessage(msg);
    }
```

```
    private class OpersysWorkerThread extends Thread{
    public OpersysWorkerThread(String name) {
        super(name);
    }

    public void run() {
        Looper.prepare();
        mHandler = new OpersysWorkerHandler();
        Looper.loop();
        }
    }

    private class OpersysWorkerHandler extends Handler {
    private static final int MESSAGE_SET = 0;

    @Override
    public void handleMessage(Message msg) {
        try {
        if (msg.what == MESSAGE_SET) {
            Log.i(TAG, "set message received: " + msg.arg1);
            }
        }
        catch (Exception e) {
        // Log, don't crash!
        Log.e(TAG, "Exception in OpersysWorkerHandler.handleMessage:", e);
            }
        }
    }
}
```

8.2. IOpersysService.aidl

```
package android.os;
/**
* {@hide}
*/
interface IOpersysService {
void setValue(int val);
}
```

8.3. frameworks/base/Android.mk

...

```
    core/java/android/os/IPowerManager.aidl \
    core/java/android/os/IOpersysService.aidl \
    core/java/android/os/IRemoteCallback.aidl \
```

...

8.4. SystemServer.java

Should eventually be Context.OPERSYS_SERVICE

```
...
            try {
                Slog.i(TAG, "Opersys Service");
                ServiceManager.addService("opersys", new OpersysService(context));
            } catch (Throwable e) {
                Slog.e(TAG, "Failure starting OpersysService Service", e);
            }
...
```

8.5. - app/ContextImpl.java

```
...
import android.os.IOpersysService;
import android.os.OpersysManager;
...
     registerService(OPERSYS_SERVICE, new ServiceFetcher() {
             public Object createService(ContextImpl ctx) {
                 IBinder b =
                   ServiceManager.getService(OPERSYS_SERVICE);
                 IOpersysService service =
                   IOpersysService.Stub.asInterface(b);
                 return new OpersysManager(service);
             }});
...
```

8.6. content/Context.java

```
...
    /**
     * Use with {@link #getSystemService} to retrieve a
     * {@link android.nfc.NfcManager} for using NFC.
     *
     * @see #getSystemService
     */
    public static final String NFC_SERVICE = "nfc";

    /** The Opersys service **/
    public static final String OPERSYS_SERVICE = "opersys";
...
```

8.7. os/OpersysManager.java

```
package android.os;

import android.os.IOpersysService;

public class OpersysManager
{
    public void setValue(int value)
    {
        try {
            mService.setValue(value);
        } catch (RemoteException e) {
        }
    }

    public String read(int maxLength) {...}

    public int write(String stringVal) {...}

    public OpersysManager(IOpersysService service)
    {
        mService = service;
    }

    IOpersysService mService;
}
```

9. Shell Scripts

- Android has built-in shell
- Since 4.0, MirBSD Korn Shell
- See: https://www.mirbsd.org
- Start shell script like any other native utility
- For example, init.goldfish.rc does:

```
service goldfish-setup /system/etc/init.goldfish.sh
    user root
    group root
    oneshot
```

10. init.rc Commands and Services

- Edit relevant init.*.rc to:
  - Add commands to existing actions
  - Create new actions based on global property triggers
  - Add new services
- See:
  - system/core/root/init.rc
  - device/[vendor]/[product]/init.*.rc

11. C Libraries

- Build libs as part of AOSP or using NDK:

  BUILD_SHARED_LIBRARY
- Use library in native utility or daemon:

  LOCAL_SHARED_LIBRARIES := libfoo ...
- Provide headers files as needed
- Can load library explicitly in Java, espc. for JNI
- See “HelloJNI” example in NDK for NDK ex.

```
public class HelloJni extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        /* Create a TextView and set its content.
         * the text is retrieved by calling a native
         * function.
         */
        TextView  tv = new TextView(this);
        tv.setText( stringFromJNI() + " " + pid() );
        setContentView(tv);
    }

    /* A native method that is implemented by the
     * 'hello-jni' native library, which is packaged
     * with this application.
     */
    public native String  stringFromJNI();
...
    /* this is used to load the 'hello-jni' library on application
     * startup. The library has already been unpacked into
     * /data/data/com.example.HelloJni/lib/libhello-jni.so at
     * installation time by the package manager.
     */
    static {
        System.loadLibrary("hello-jni");
    }
}
```

12. Java Libraries

- Build as part of AOSP:

  BUILD_JAVA_LIBRARY
- Or create using Eclipse
- Will generate a .jar
- See:
  - http://www.vogella.com/articles/AndroidLibraryProjects/article.html
  - http://stackoverflow.com/questions/3642928/adding-a-library-jar-to-an-eclipse-android-project

13. SDK add-ons

- See device/sample/, espc.:
  - README
  - products/sample_addon.mk
  - sdk_addon/
  - frameworks/PlatformLibrary/
- Creates a ZIP file
- Extract in [sdk-dir]/add-ons/

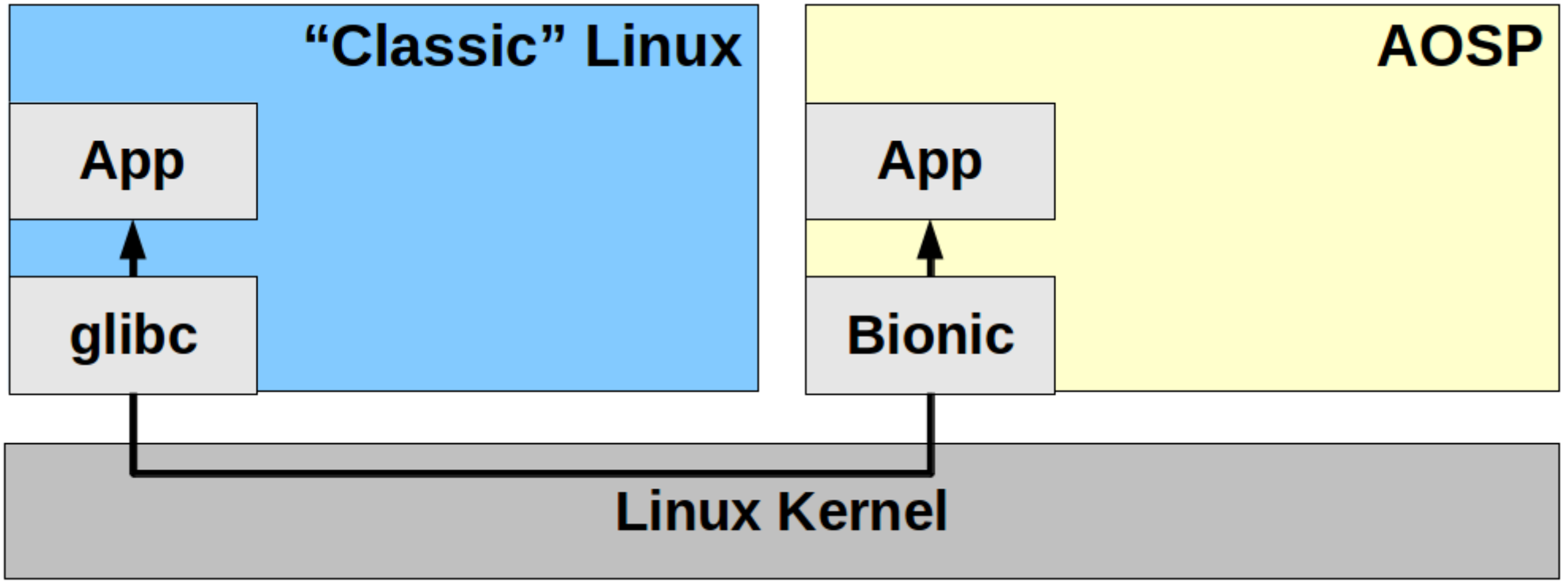
14. Legacy Linux

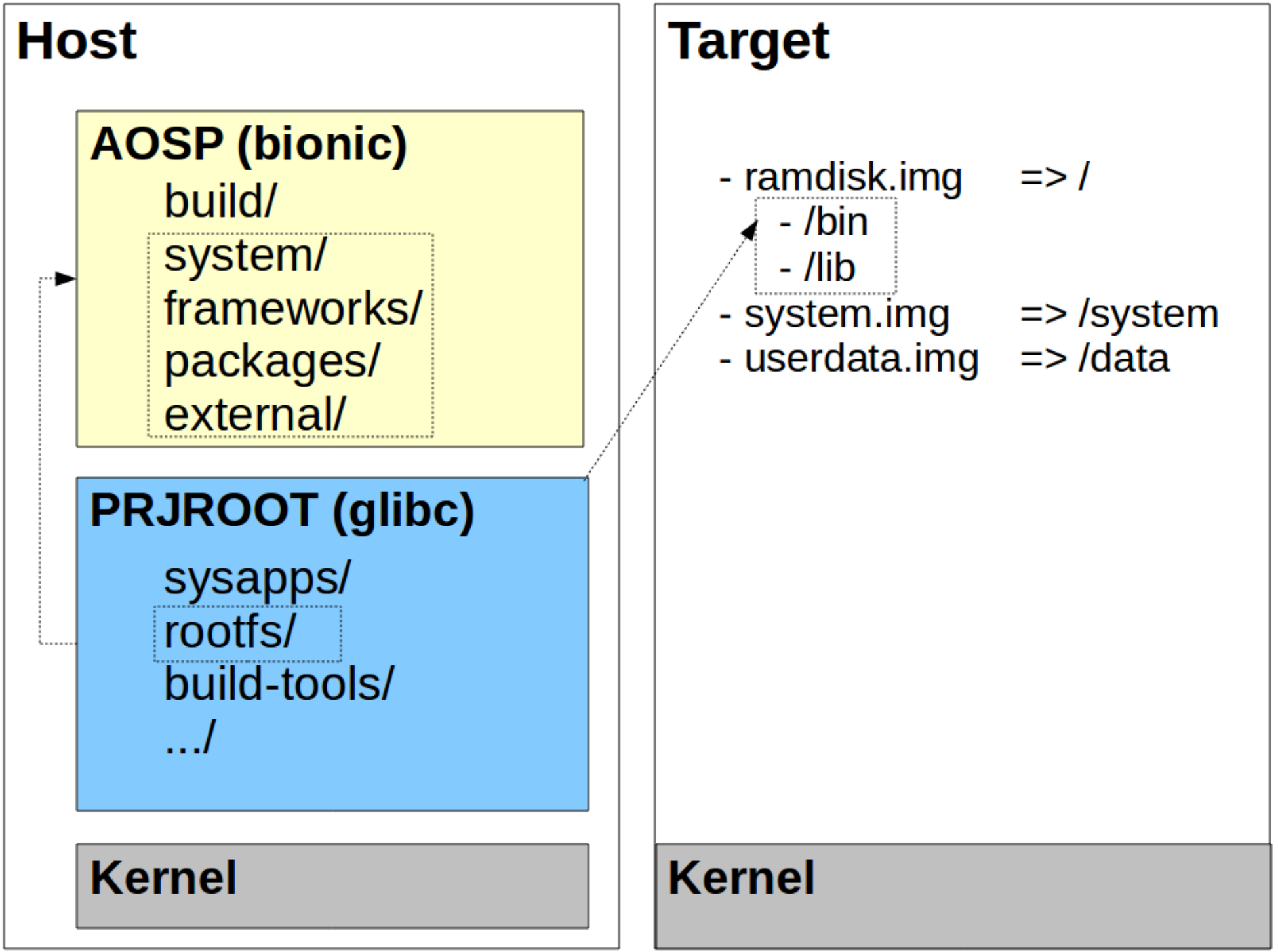
- Why?
- Architecture
- Build integration

14.1. Why?

- What if I ... ?:
  - have been using my custom rootfs forever
  - really need glibc
  - have a huge stack based on “legacy Linux”
  - don't want to deal with AOSP's build system
  - want BusyBox
  - want to unify with a std Linux desktop
  - need things the AOSP doesn't provide
  - have a ton of Linux drivers and want those to be used by Android
  - ...
- Know this: filesystem integration is trivial:
  - Change build system to copy your custom rootfs
  - Change init scripts to start you own daemons
  - Use sockets to communicate
- Know this too: std Linux desktop integration is NOT

14.2. Architecture




Host
AOSP (bionic)
build/
system/
frameworks/
packages/
external/
PRJROOT (glibc)
sysapps/
rootfs/
build-tools/
.../
Kernel
Target
- ramdisk.img => /
- /bin
- /lib
- system.img => /system
- userdata.img => /data
Kernel

14.3. Build integration

- Edit system/core/include/private/android_filesystem_config.h and add “/lib”
- Modify build system to copy your tree to final rootfs
- Change init.rc to:
  - Start your “services”
  - Make BusyBox the default console
- Change adb to start BusyBox shell instead of default shell

Thank you ...

karim.yaghmour@opersys.com