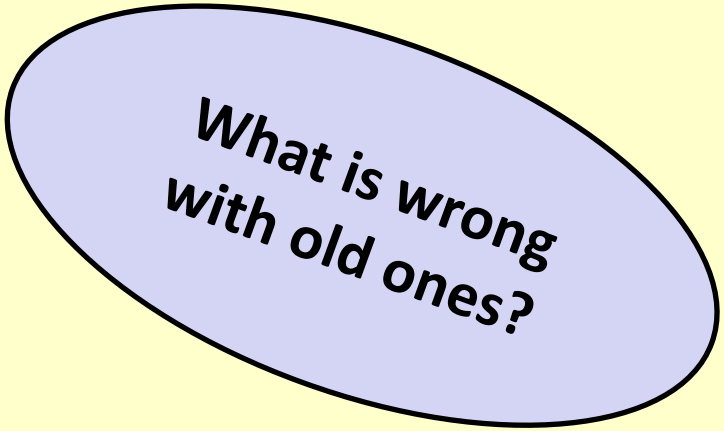
Android (and iOS)

Professor Hugh C. Lauer
CS-3013 — Operating Systems

Materials for this topic

- **Based on CS-502 projects:–**
- ***Traditional* vs. *Mobile Operating Systems***
  - Jeffrey Martin, James Process, Mike Sandman, and Andrew Sawchuk, CISCO SYSTEMS, Fall 2011 (report(pdf), slides (pdf)
- ***Android***
  - Kathy Wienhold, iRobot, Summer 2010 (report(pdf), slides (pdf))
- **Lots of material on the web and Wikipedia**
  - Wikipedia article is more metadata than content!

New Age of Operating Systems



- End user expectations
- Simpler, cleaner design for limited peripherals
- Reliability considerations for mobile devices

Design

- **Android based on Linux but Android is not Linux!**
  - Aligned with recent Linux kernel
  - Many “patches”
  - Totally different programming environment

- **iOS is based on MacOS but iOS is not MacOS!**
  - (relationship to MacOS = ???)
  - Totally different programming environment

Comment on Programming Environments

- **Programming a GUI is vastly different from programming for a command-line-based system**
  - GUI techniques originated at SRI and Xerox PARC
  - Macintosh adopted
  - Apple evangelization effort!
  - Windows did not “get it”
  - …. until Windows 95/NT

- **Programming a mobile/touch-screen device is different from programming for a GUI**

Application Programming — Android

- **Languages**
  - Java!
  - Other languages possible thru JNI
    - All system APIs must be called from Java code

- **Integrated Development Environment (IDE)**
  - SDK based on Eclipse
  - Emulation and support tools in Eclipse

Application Programming — iOS

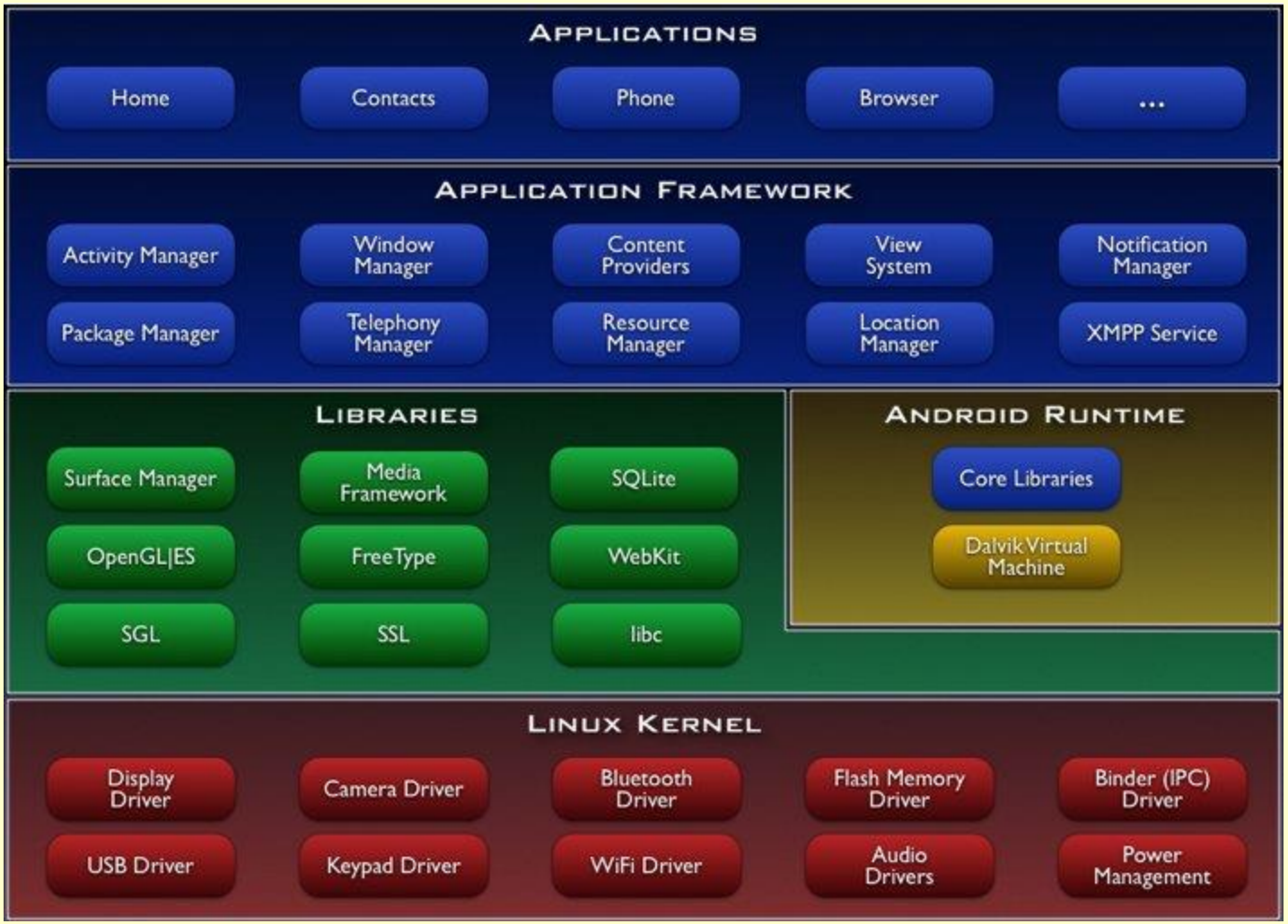
- **Languages**
  - Objective-C!
  - Library support thru Frameworks
- **Integrated Development Environment (IDE)**
  - Xcode
  - Same platform for developing MacOS applications

Both Android & iOS:

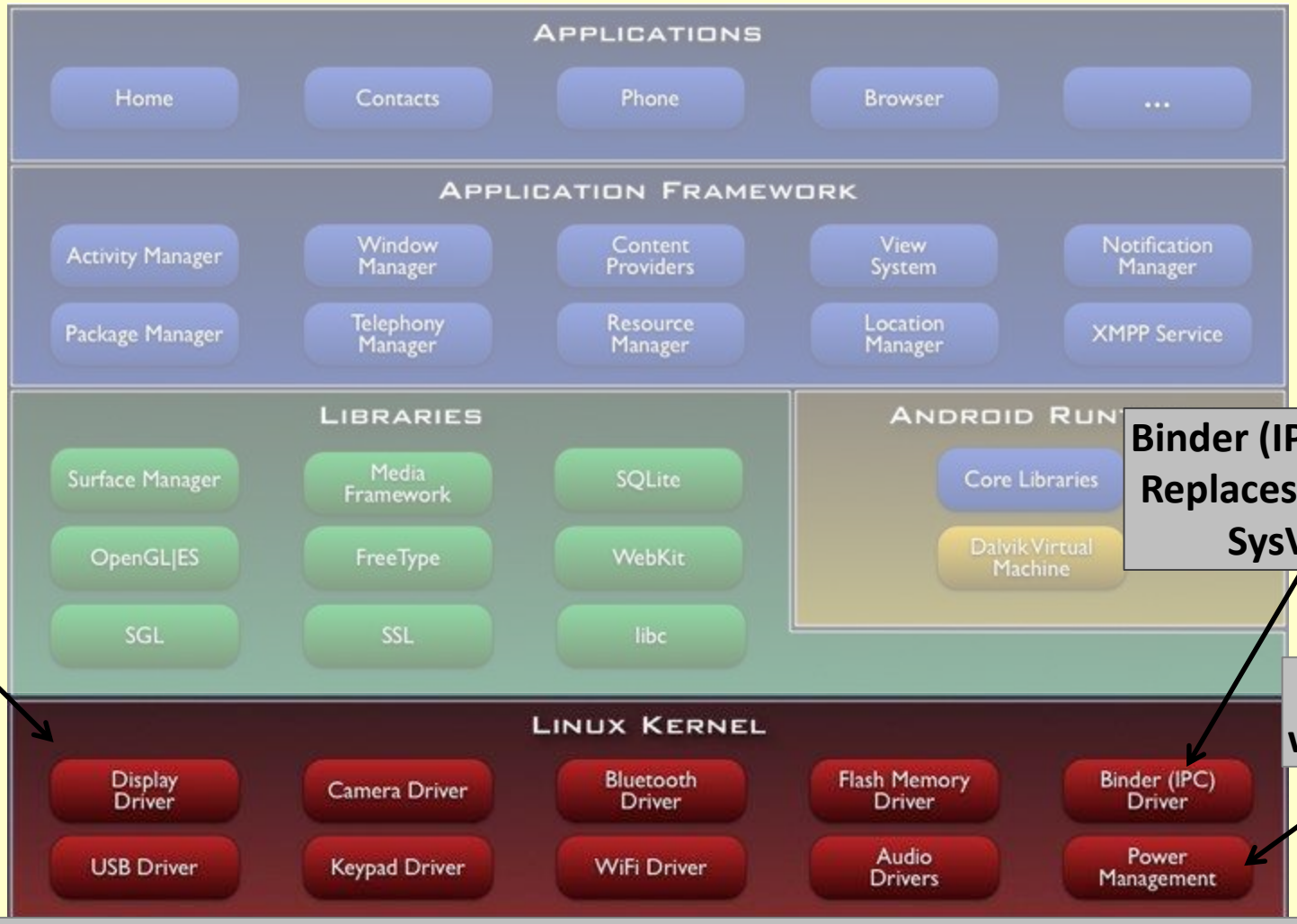
- **Interface builder**
- **Simulator**
- **Tracing & profiling tools**

Questions?

Android Architecture



Android Architecture: Linux Kernel



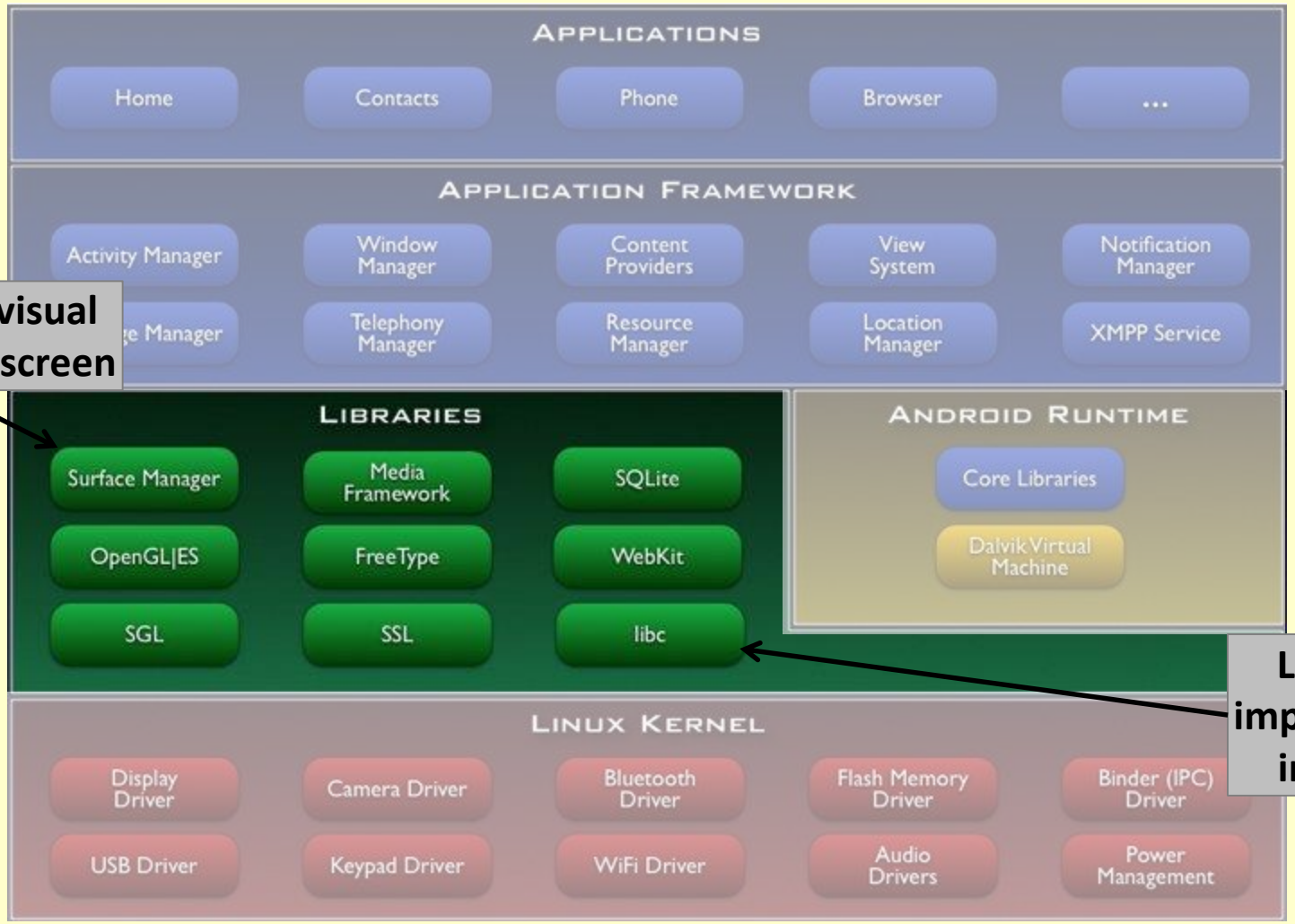
Linux: 4.4 Kernel for Android 8.1 (Oreo)

Binder (IPC) Driver: Replaces standard SysV IPC

Includes wakelocks

Hardware Abstraction Layer: Driver Model (& Drivers); Memory Management; Security Model; Process Management; Networking

Android Architecture: Libraries

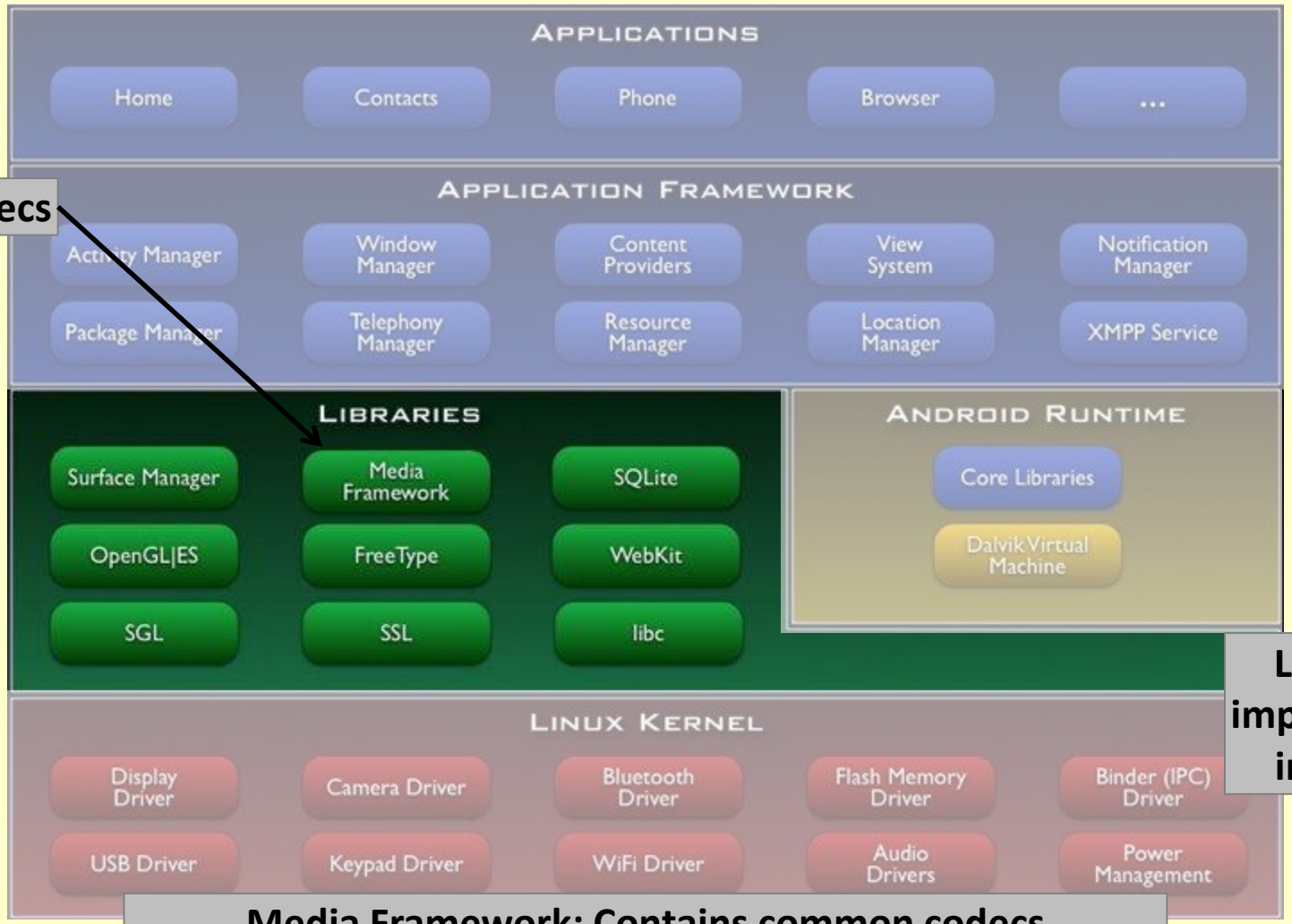


Composing visual elements on screen

Libraries implemented in C/C++

Surface Manager: Composes drawing surfaces (from different processes) onto the screen.

Android Architecture: Libraries

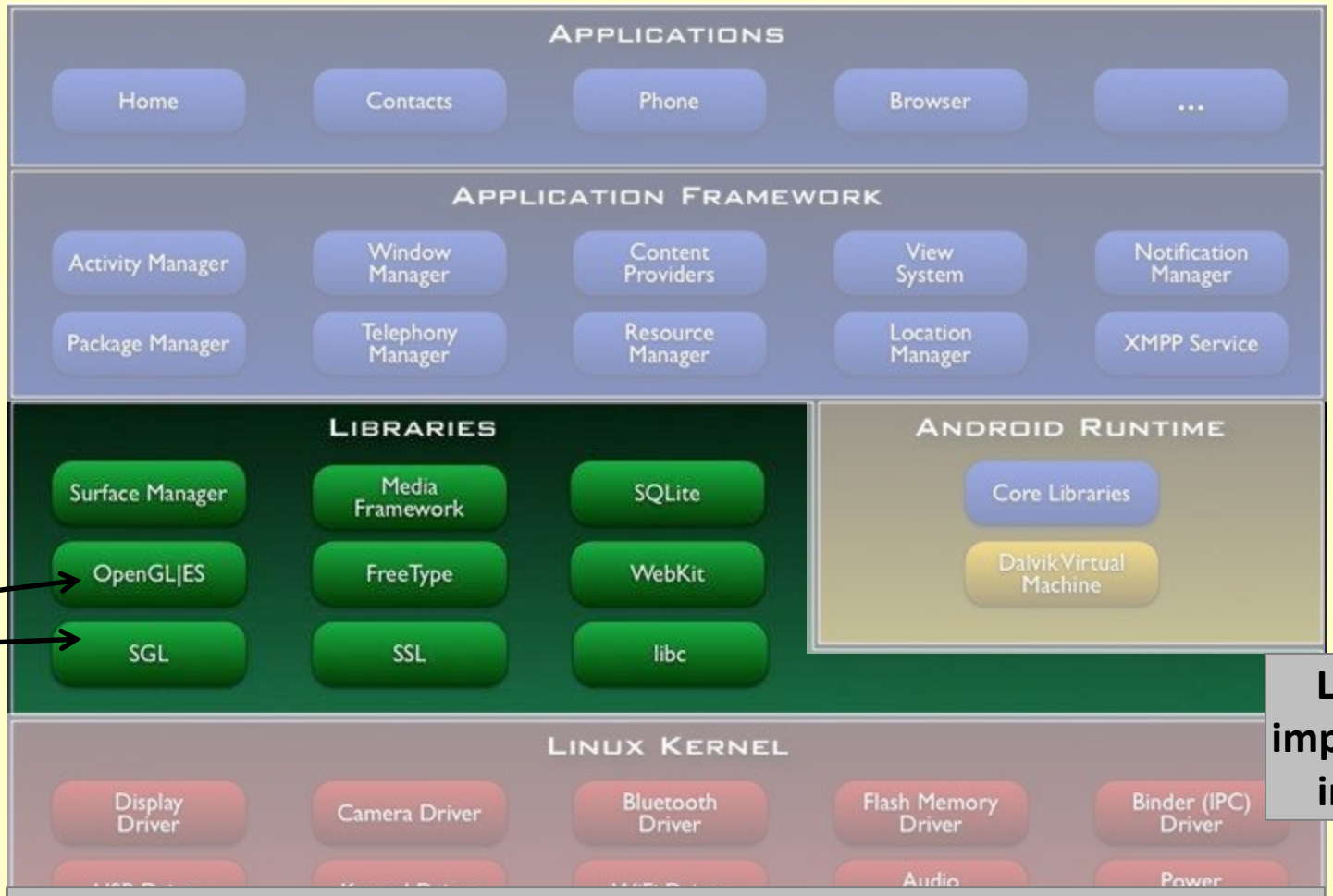


Media Codecs

Libraries implemented in C/C++

Media Framework: Contains common codecs (video & audio (MPG-4, H.264, MP3, AAC, etc.)); Provided by Packet Video (part of Open Handset Alliance)

Android Architecture: Libraries



Graphics library

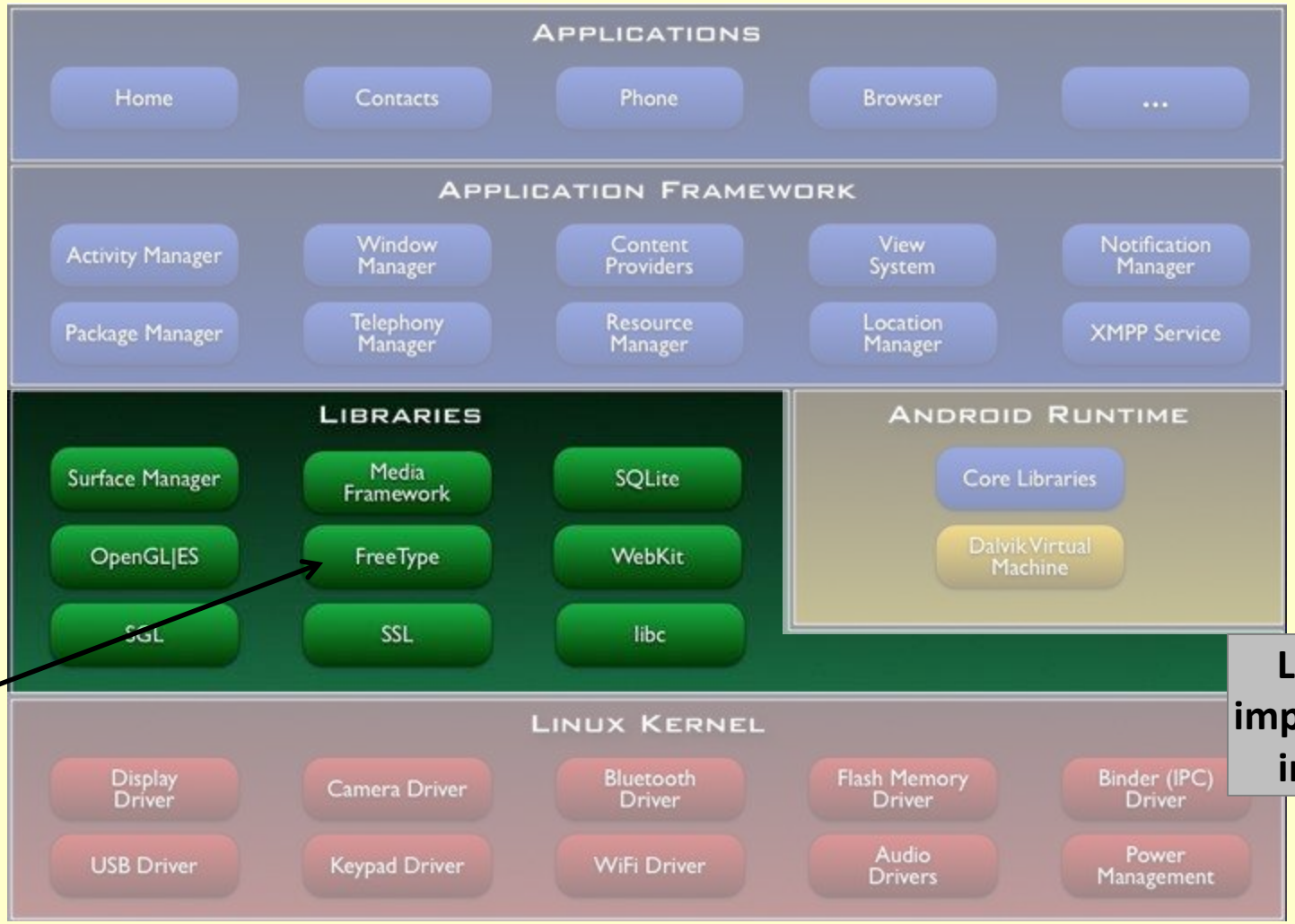
3D

2D

Libraries implemented in C/C++

Graphics Libraries (OpenGL | ES; SGL): 3D & 2D graphics packages (3D is software implementation that is HW-accelerable, if supported); 3D & 2D can be used in same application

Android Architecture: Libraries

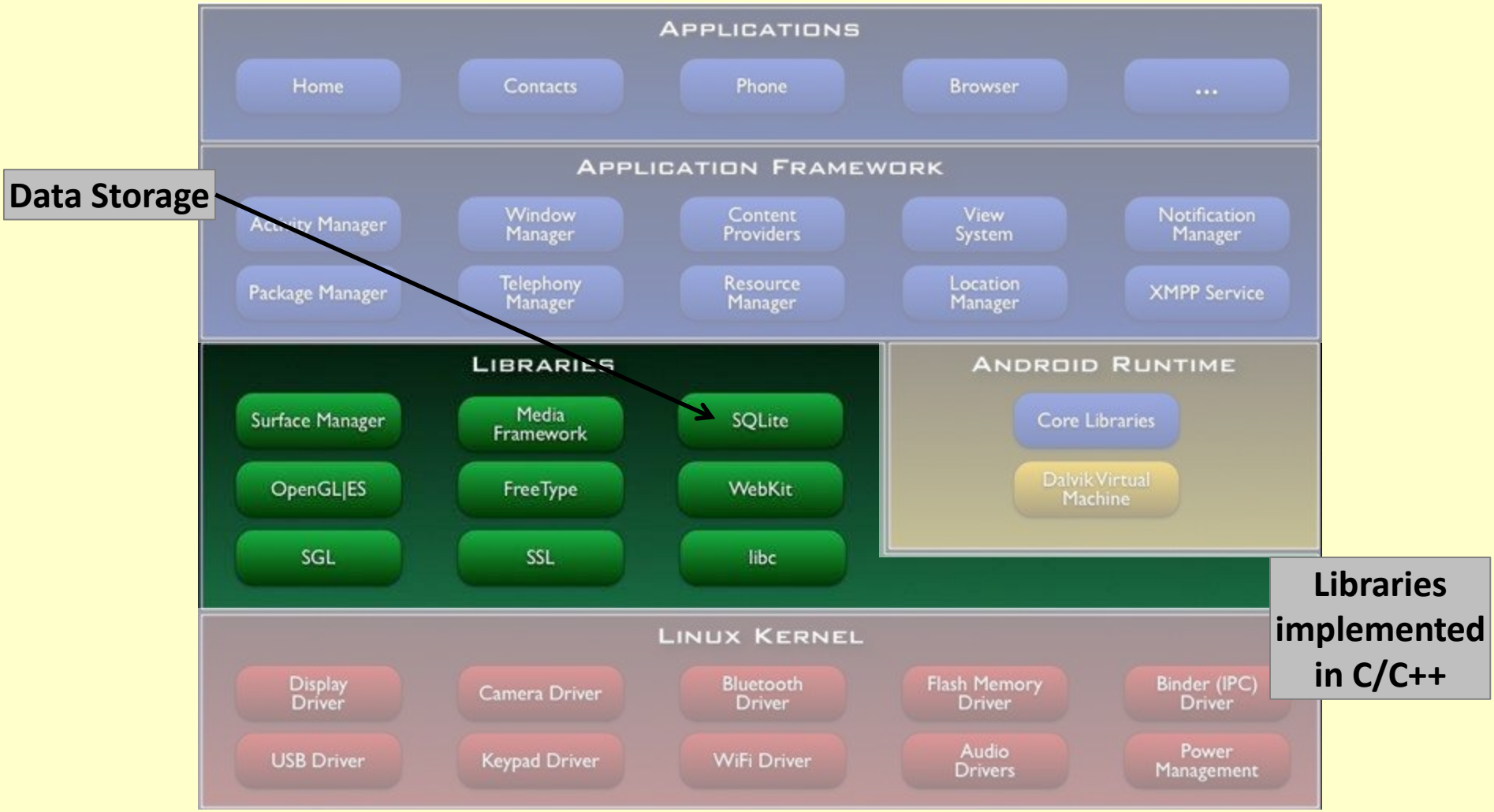


Fonts

Libraries implemented in C/C++

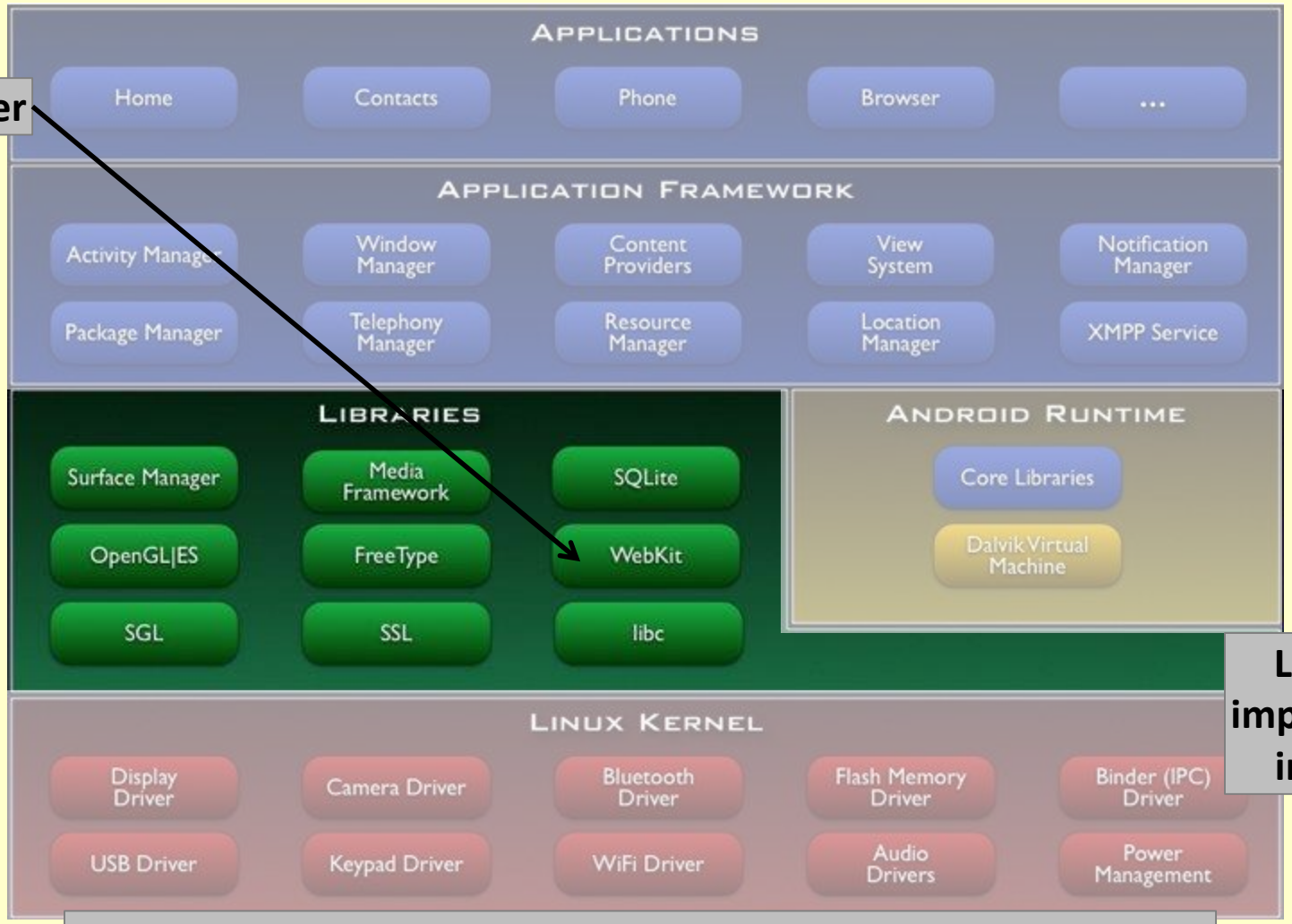
FreeType: Font rendering

Android Architecture: Libraries



SQLite: Data storage

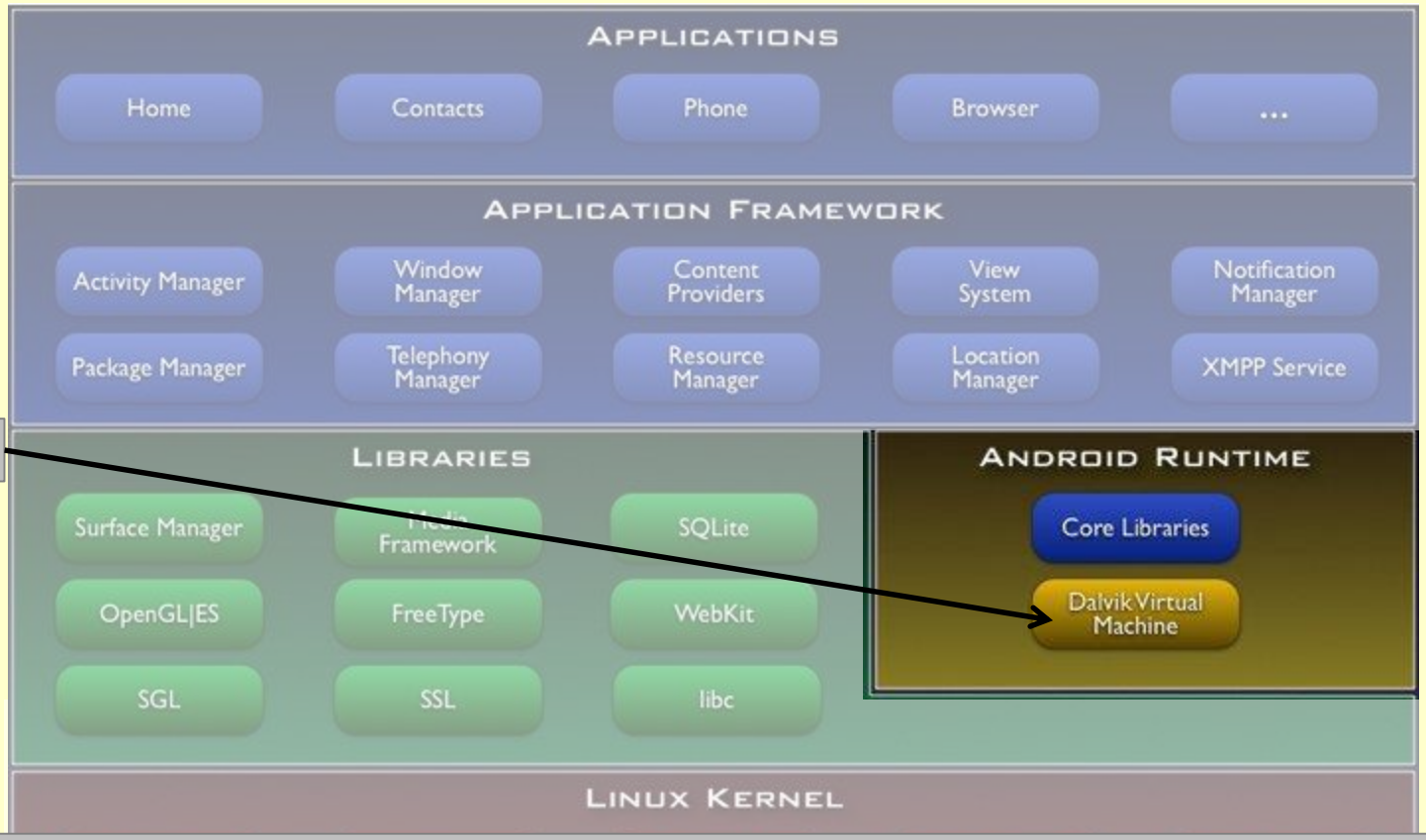
Android Architecture: Libraries



Libraries implemented in C/C++

WebKit: Open source browser engine (also powers Safari); adapted to render well on small screens used by mobile devices

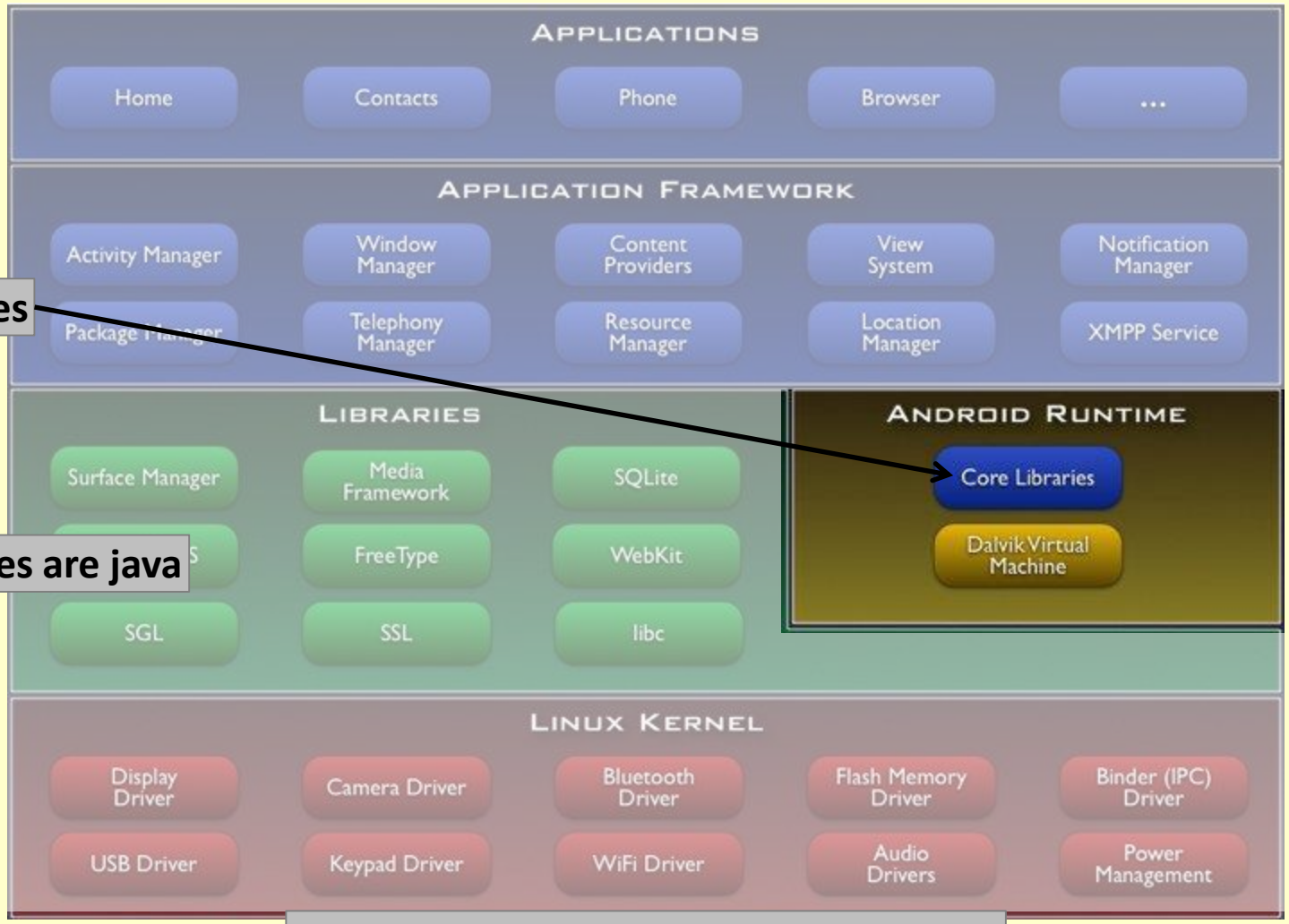
Android Architecture: Android Runtime



Dalvik VM

Dalvik VM:
Adapted from Java (modified for efficient memory/power/CPU);
usage (data structures can be shared across processes, where possible, optimized));
runs .dex file (byte codes built at compile-time);
multiple instances of VM can run at same time

Android Architecture: Android Runtime

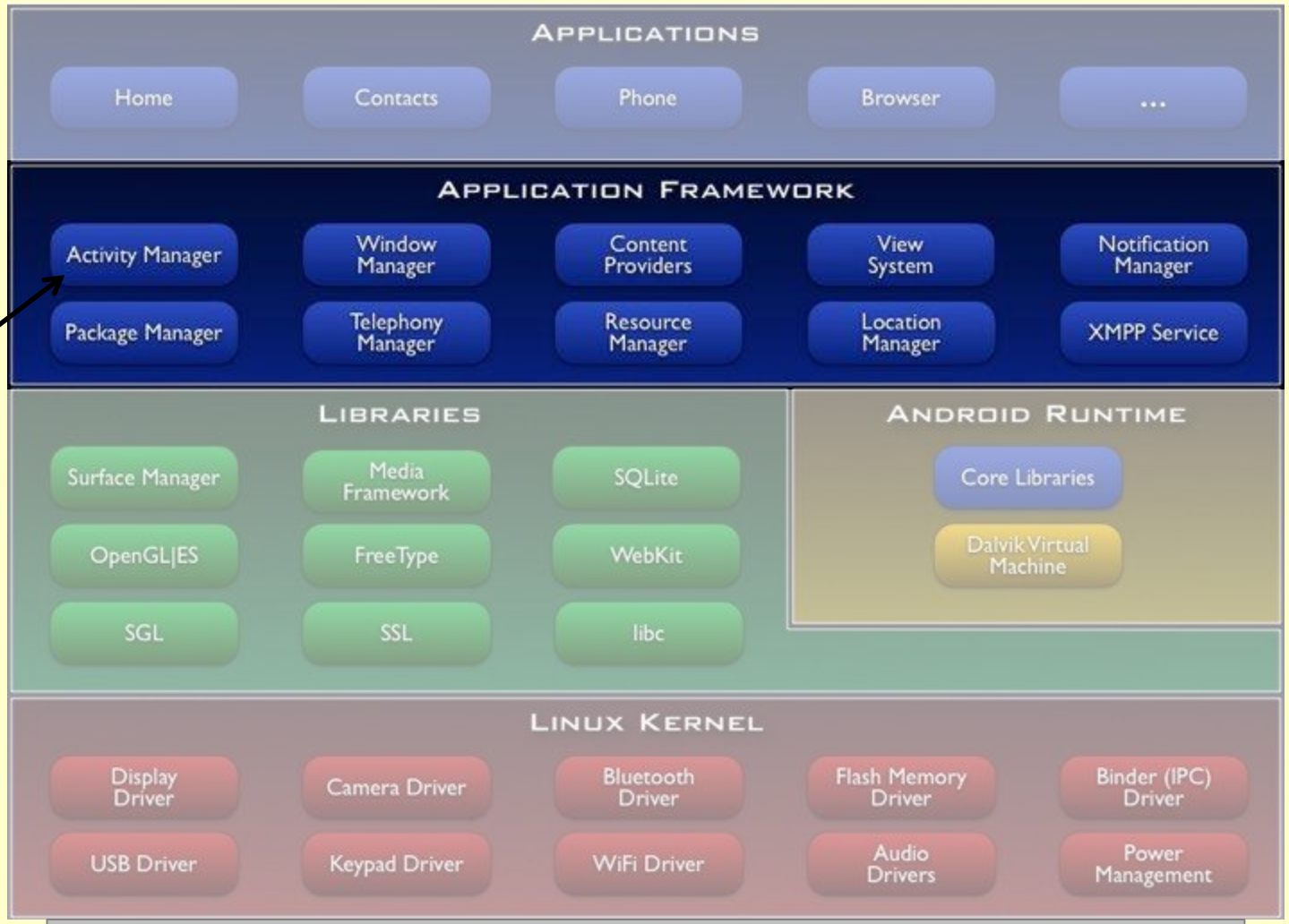


Core Libraries

All blue boxes are java

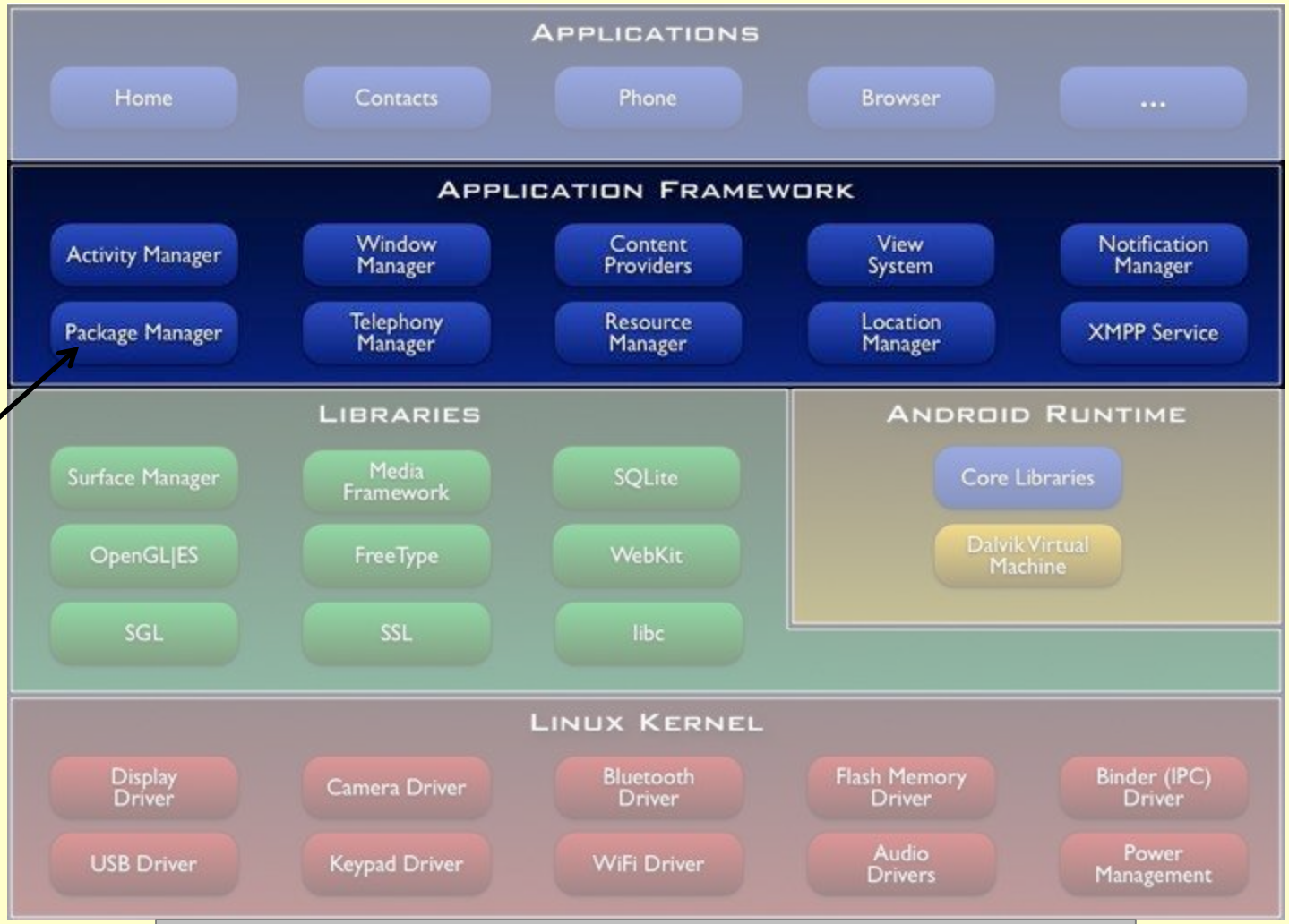
Core Libraries: Written in java; garbage collection; classes; utilities; I/O

Android: Application Framework



Activity Manager: Manages life cycle of applications; maintains a common "back" stack (smooth navigation experience)

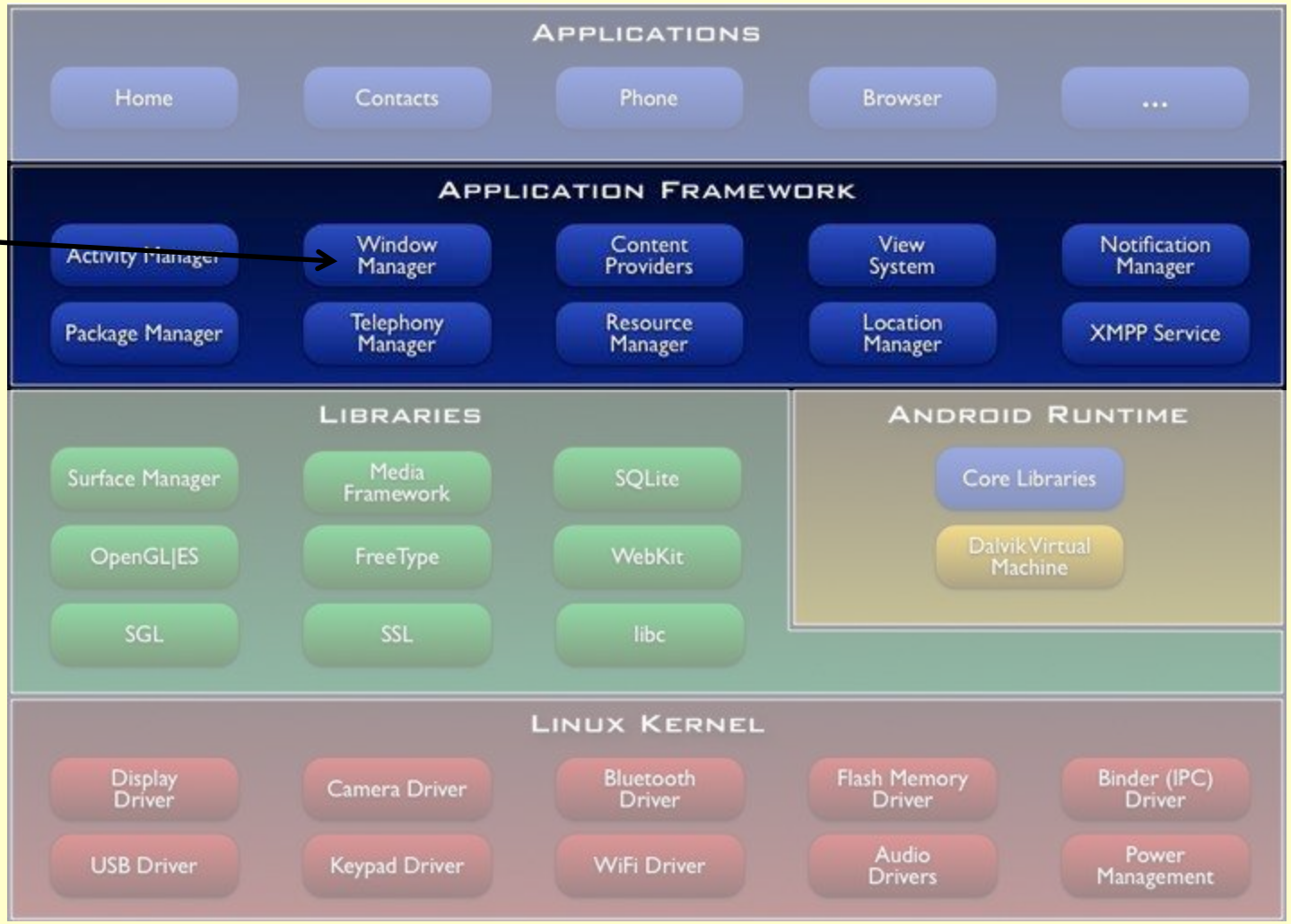
Android: Application Framework



Package Manager

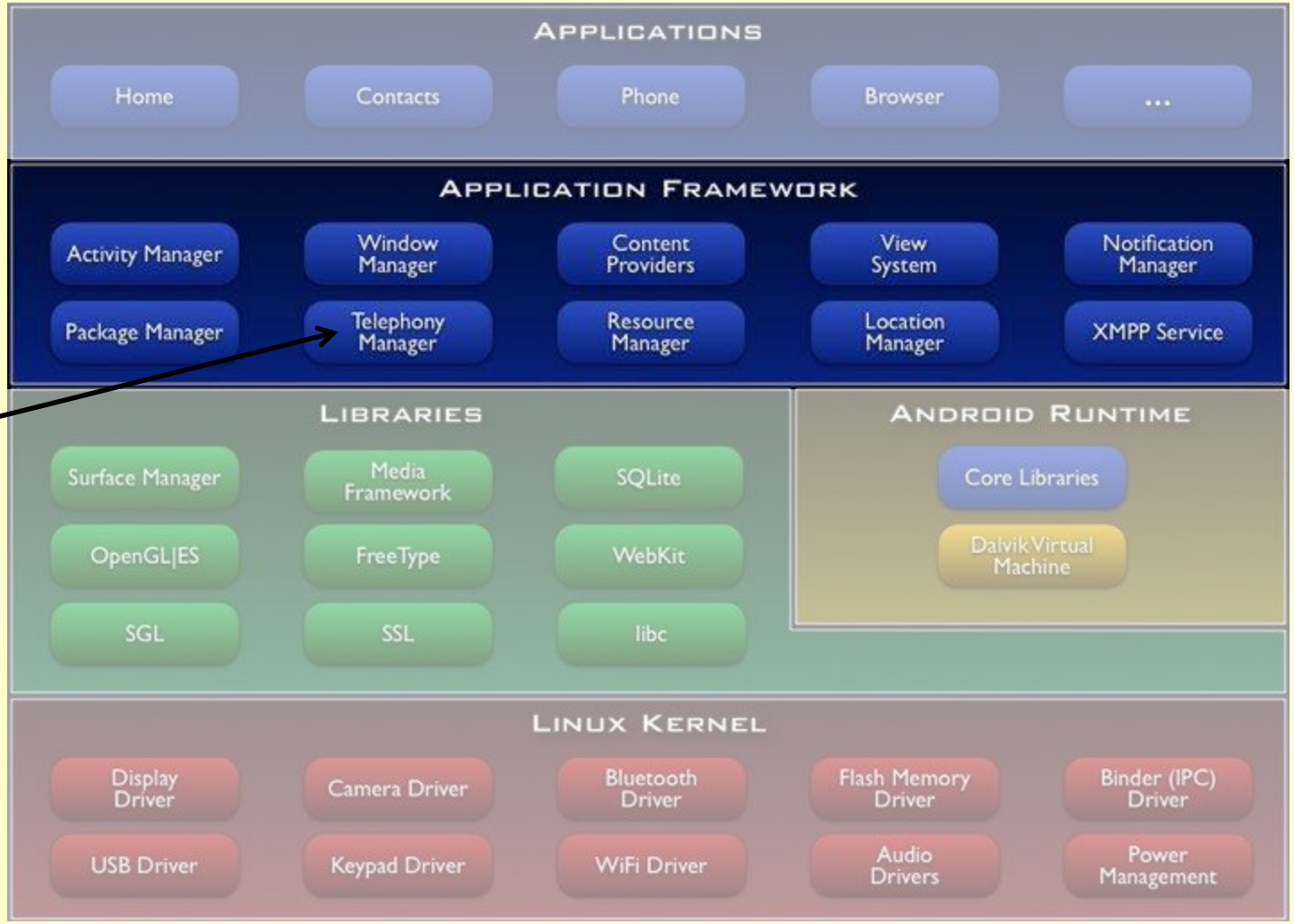
Package Manager: Keeps track of which applications are installed on device, and what their capabilities are

Android: Application Framework



Window Manager: Manages windows; builds on Surface Manager;

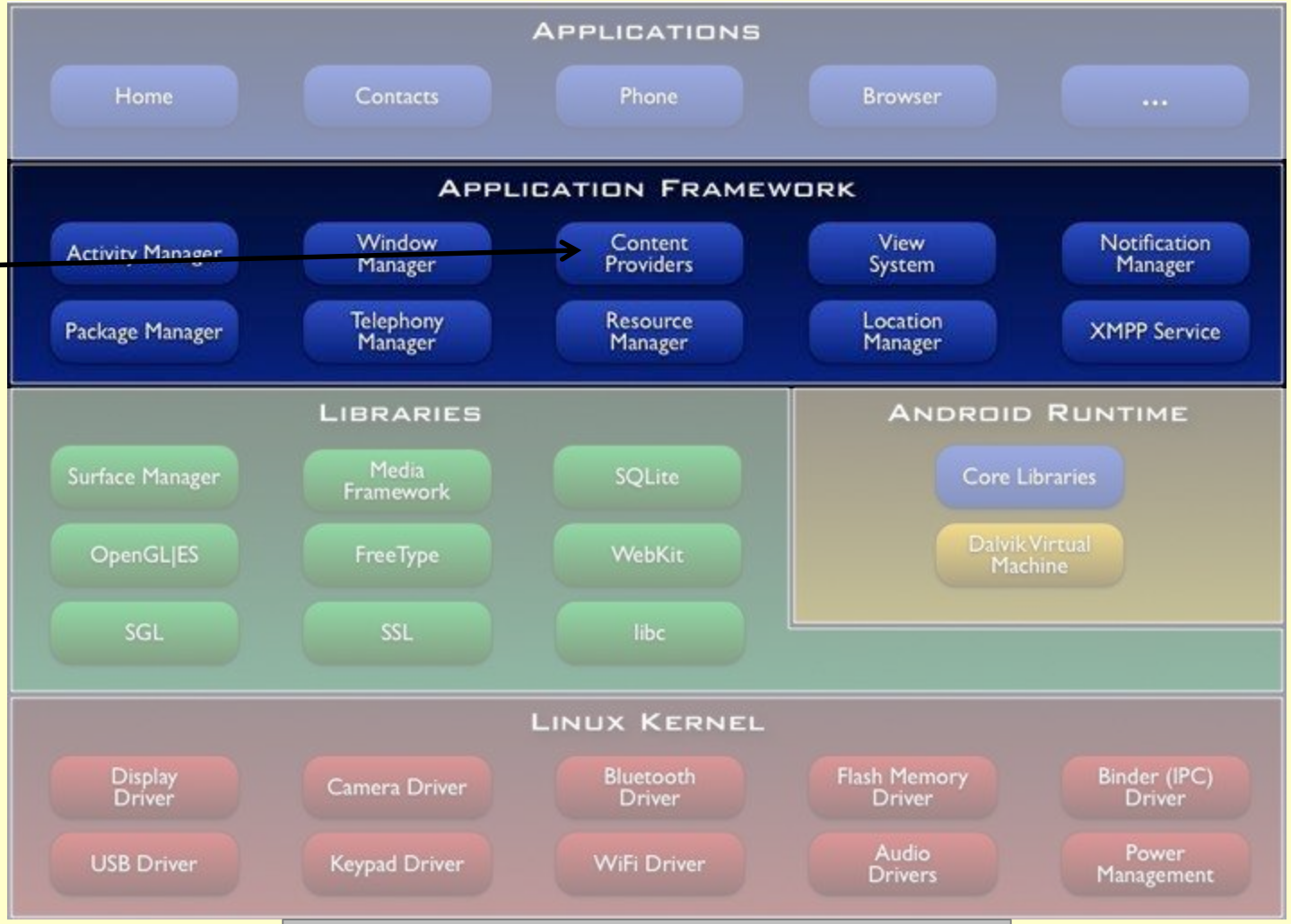
Android: Application Framework



Telephony Manager

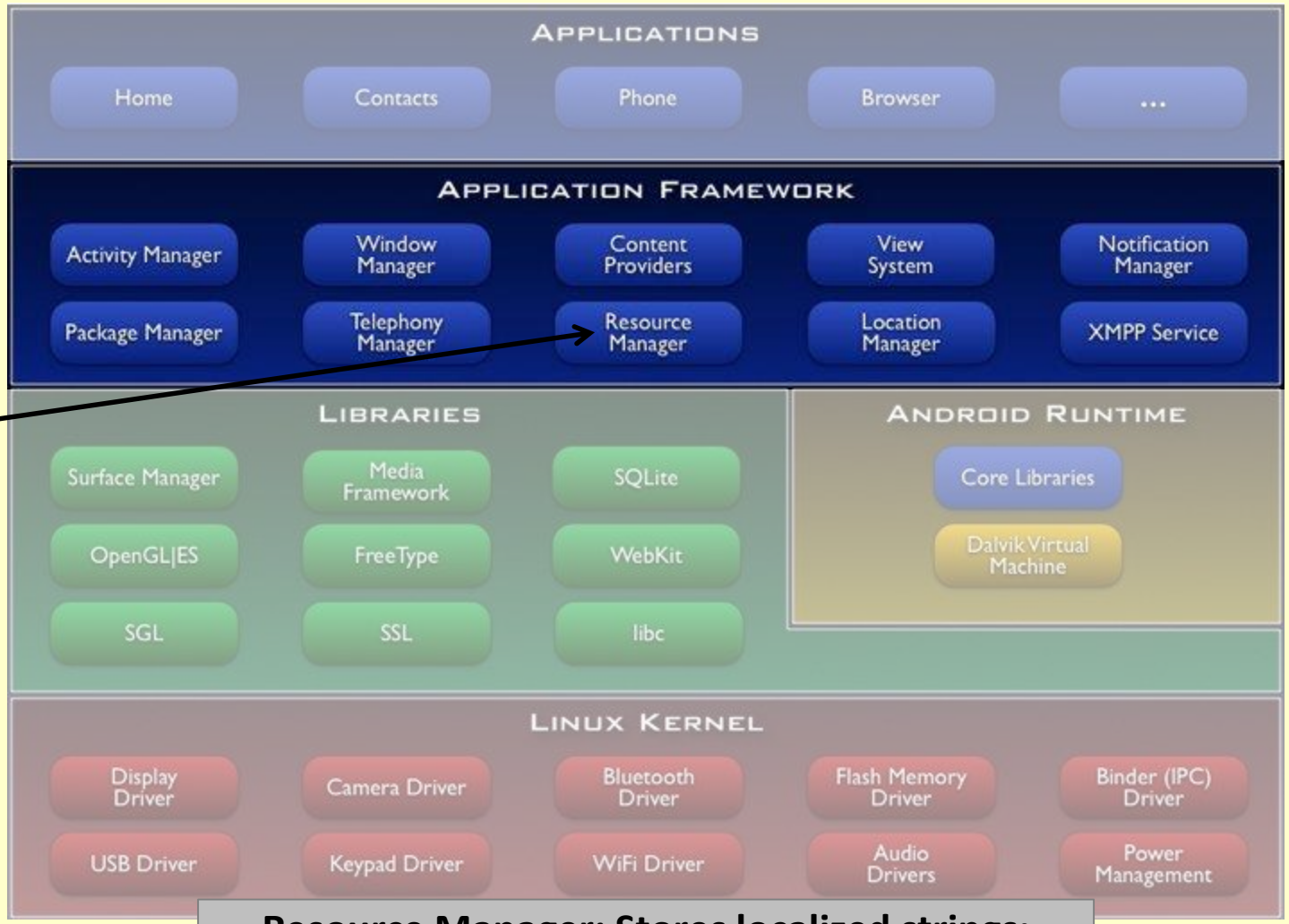
Telephony Manager: Contains API for phone application

Android: Application Framework



Content Providers: Used to share data among applications (example: Contacts)

Android: Application Framework



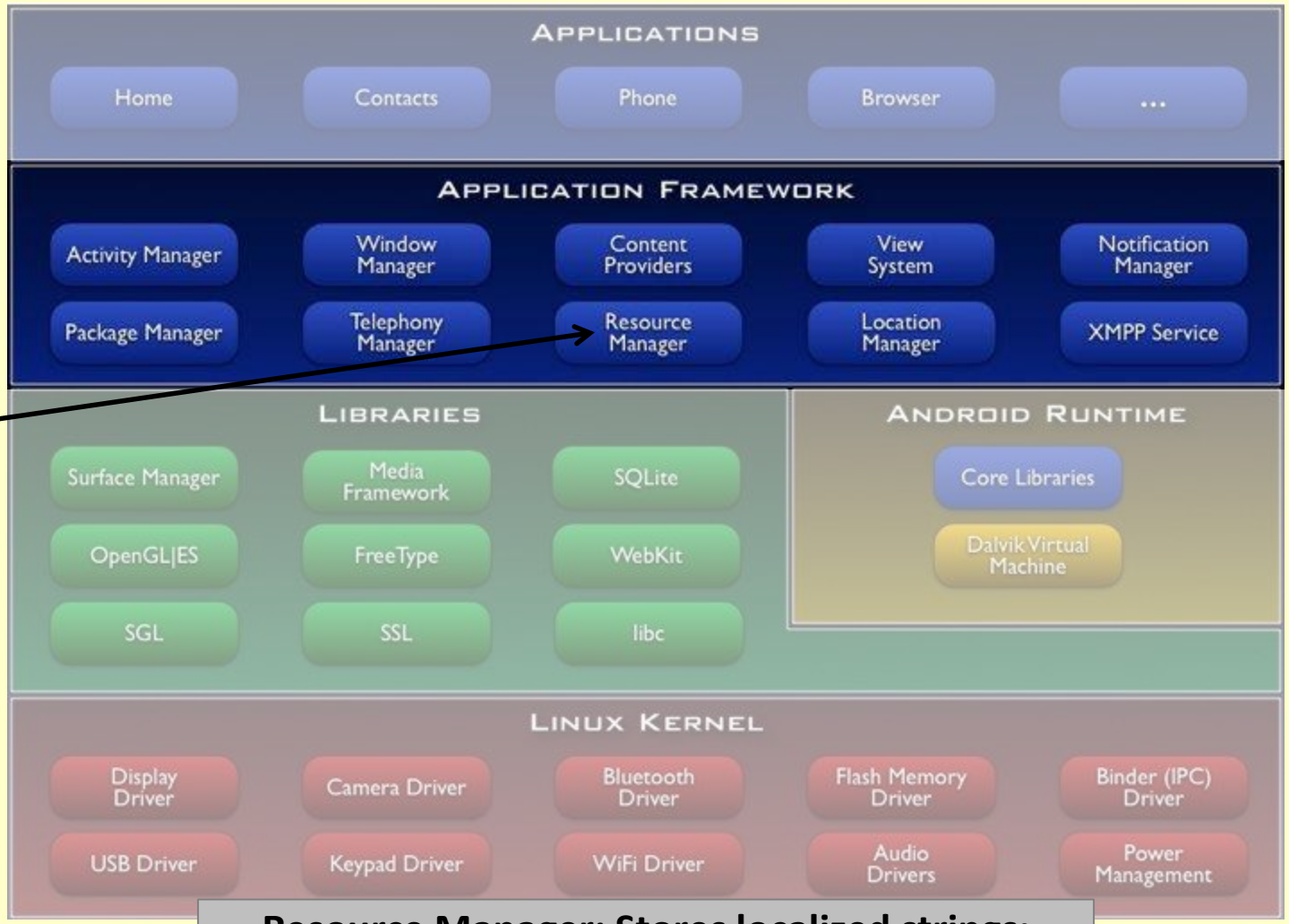
Resource Manager

Resource Manager: Stores localized strings; bitmaps; layout file descriptions (external parts of an application that aren't code)

Digression: Resource Manager

- **Consider an application designed for markets in multiple languages**
- **How do you handle**
  - Names of menu items?
  - Labels, instructions, etc?
  - Error messages?
  - Other strings
- **Answer:–**
  - The *Resource Fork*
  - A segregated area for string data
  - Separate version for each language

Android: Application Framework



Resource Manager

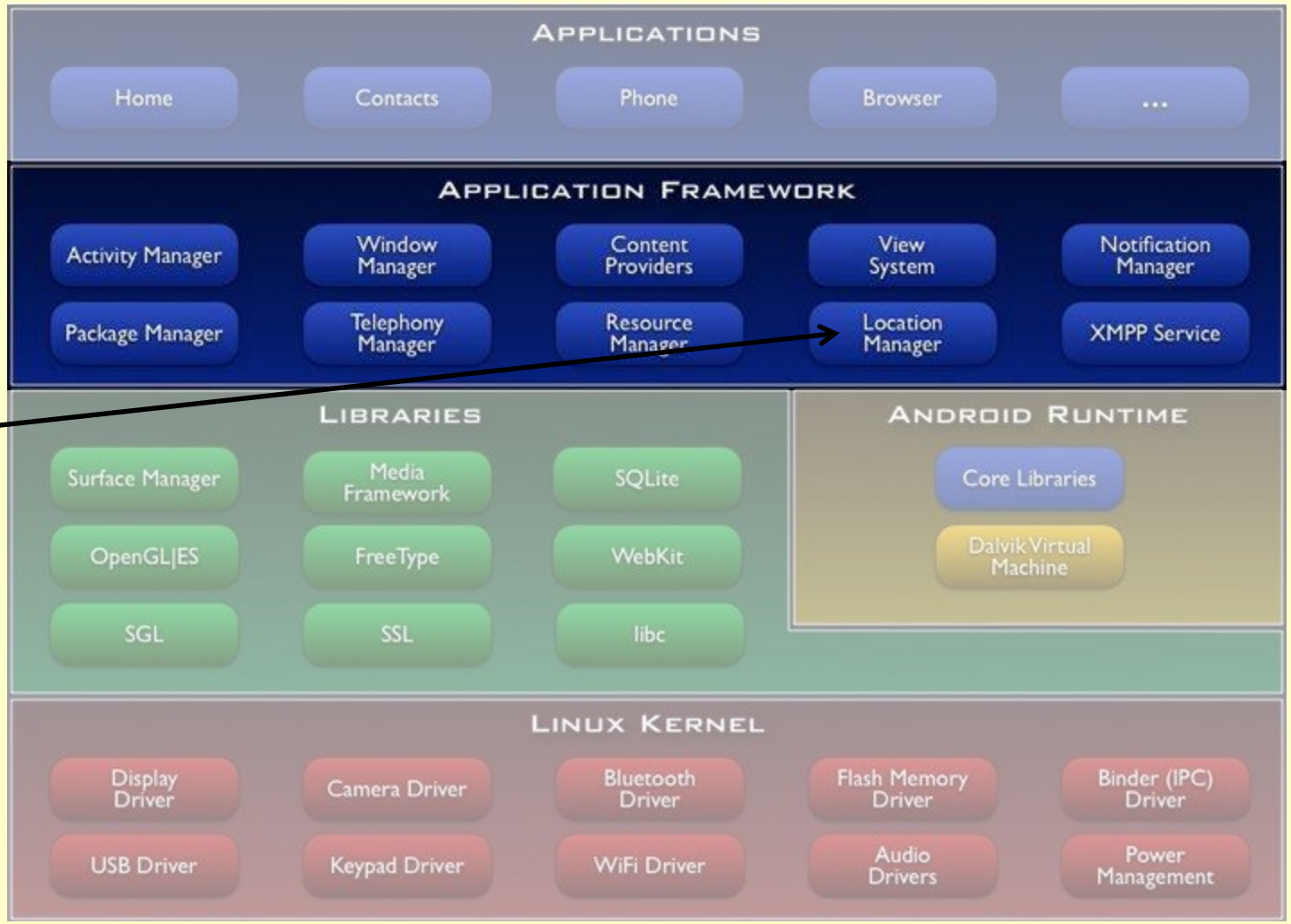
Resource Manager: Stores localized strings; bitmaps; layout file descriptions (external parts of an application that aren't code)

Android: Application Framework

View System

View System: Buttons, lists, etc. (building blocks of the UI); handles even dispatching, layout, drawing

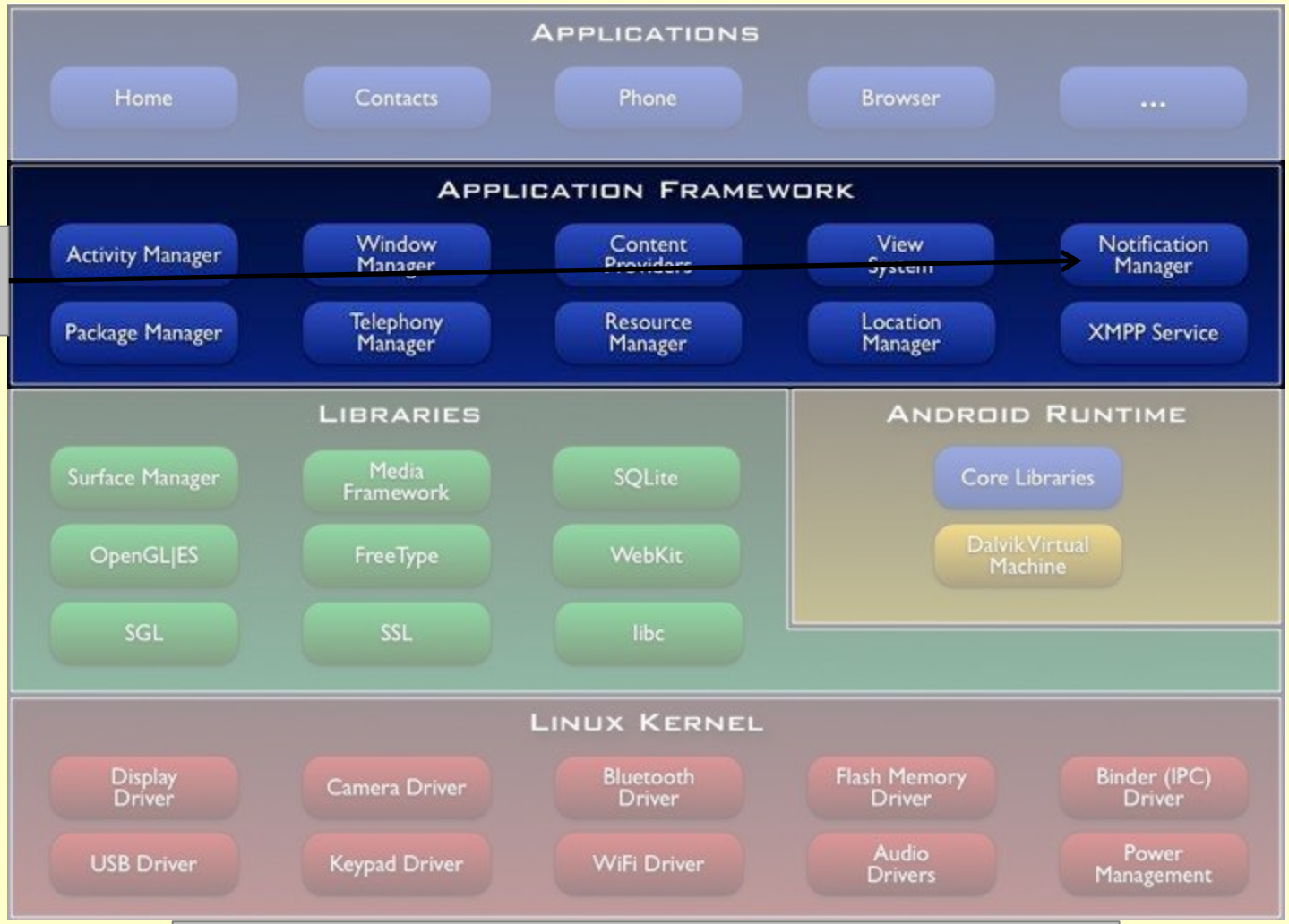
Android: Application Framework



Location Manager

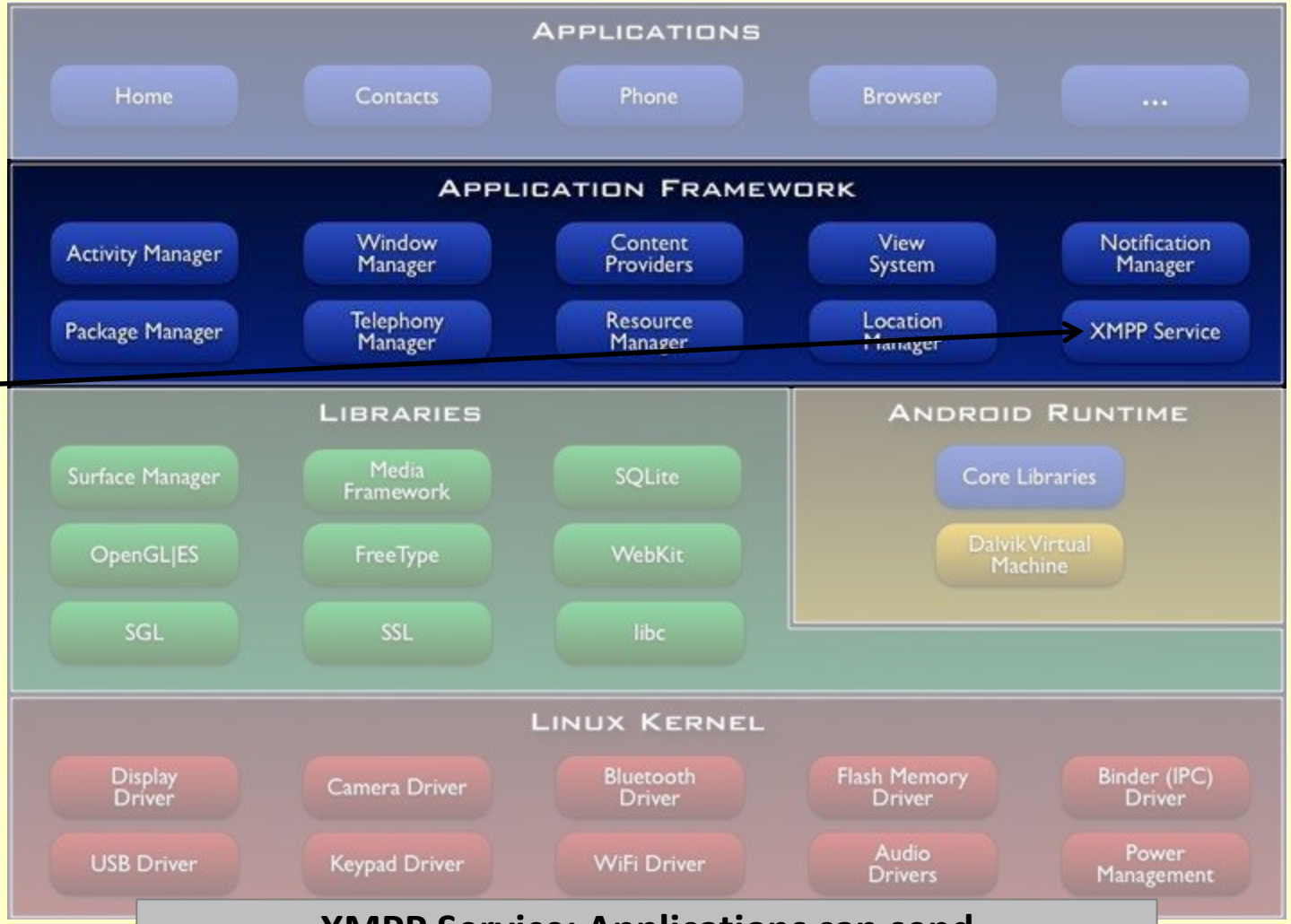
Location Manager: Applications can register with LM to be notified when sufficiently close to a location, person, etc.

Android: Application Framework



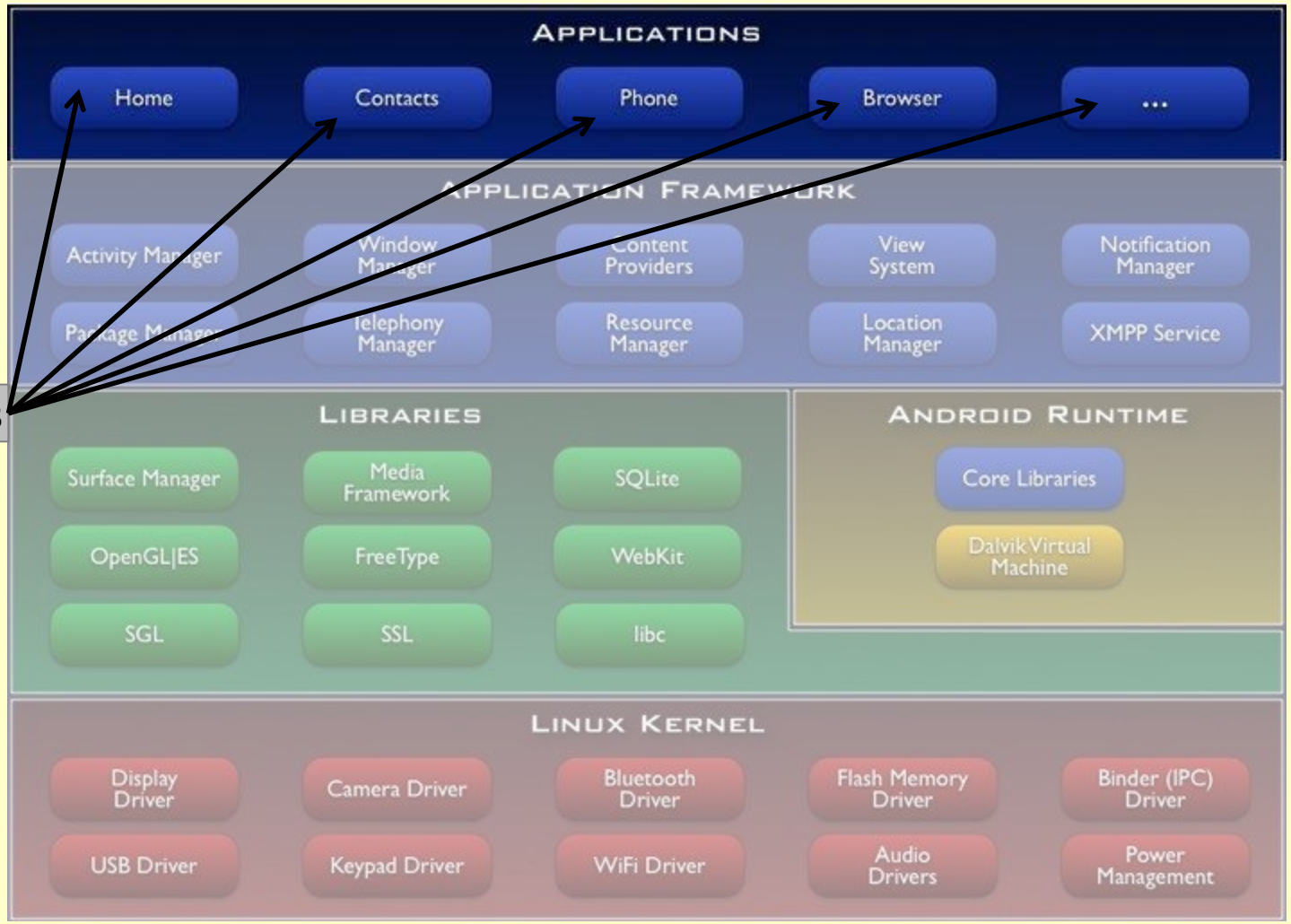
Notification Manager: Applications can be put into the status bar – alert users to interesting events

Android: Application Framework



XMPP Service: Applications can send device-to-device data messages to other Android devices (includes permission & security control)

Android Architecture: Applications



Applications: For any particular product, may be built-in or user-added

Android: Application Architecture

Application Building Blocks	
Activity	UI component typically corresponding to one screen.
Intents & Intent Receivers	Abstraction for a user-desired action that one or more components may be able to supply; registered with system via the application's manifest file – example intents: Show location, Send email
Service	Faceless task that runs in the background.
Content Provider	Enables applications to share data – example: Contacts

Basics: All applications run in their own processes. Android system itself starts and shuts down processes (resource management), transparently, including saving process state so it can be restored later. An application is typically one or more activities, plus a Linux process to contain them.

Intents: Formal class in Android system. Uses the Binder IPC functionality (to be discussed later).

Questions?

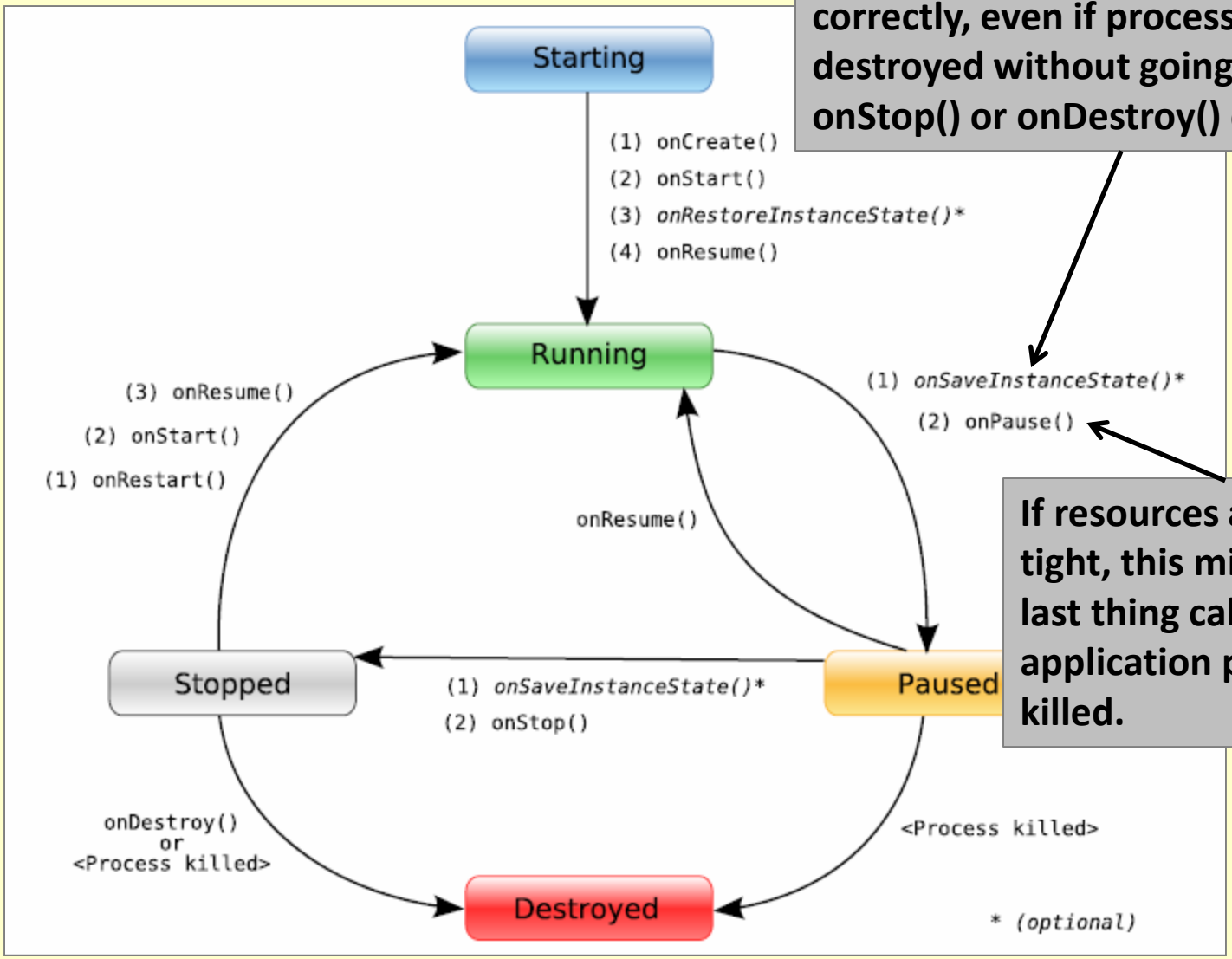
Process and Memory Management

- **Each Android application runs in own Dalvik VM**
  - Dalvik VM is memory-optimized so that multiple instances may be run on the same device
  - Threading and low-level memory management is done by the Linux kernel

- **Android Runtime**
  - Manages processes and memory at a higher level
  - Each process is assigned a state (and associated priority)
  - Android runtime kills tasks to free up memory based on priority of task

Android: Activity Lifecycle

So very good "code hygiene" to make sure this is implemented correctly, even if process is destroyed without going through onStop() or onDestroy() calls.



If resources are really tight, this might be the last thing called before application process is killed.

Image from "Hello, Android", by Ed Burnett, 3rd edition, Figure 2-3.

Android: Intents

- **"An intent is an abstract description of an operation to be performed."[1]**
- **This allows run-time binding of services and functions**

- **Parameters:**
  - action: flag for type of action
    - Examples: ACTION_VIEW, ACTION_EDIT, ACTION_MAIN (for the application's "main"), …
  - data: data to perform the action on, specified as a URI (uniform resource indicated)
    - Examples: http://www.google.com, tel:123, content://contacts/people

[1] Android Developers website, Reference, Intents

Android: Services

- Long-running application
- No direction interaction with the user
- No specific functionality for other applications to use.
- May stand-alone, or bundled with other components.
- Pretty much what you would expect a service to be.

[1] Android Developers website, Reference, Service

Android: Content Providers

- **Data accessible to all applications**
  - Example: Contacts:
  - import android.provider.Contacts.People;
- **Database-style interface**
  - Each row is record
  - Each column has a particular meaning
- **Content providers are added to application's *manifest file***
  - Announcing their presence
  - Enabling use by other applications.

Principal Abstractions & Services

1. Processes and concurrency
2. Synchronization and interprocess communication
3. Memory management, virtual memory, etc.
4. File systems and/or persistent storage
5. I/O and (possibly) graphics
6. Program loading & unloading
7. Networking and communication
8. Security
9. Multiprocessor Support

We'll look at each subject area in a separate slide, to make sure the bases are covered.

Processes & Concurrency

- **Applications are a stack, starting with the "Home" application**
- **One "foreground" application**
  - The Activity Manager keeps track of things, killing and restoring processes "behind the scenes", as needed.
- **Java threads possible, *but* …**
  - Prefer higher-level abstractions, such as the *java.util.concurrent* package.
- **See also the later discussion of Binder**

Synchronization and IPC

- **Prefer the inherent event framework**
  - onXXX(…) methods.
- **However, if exclusionary mechanisms are required, there are supporting java classes:**
  - java.util.concurrent.locks.Lock
  - java.util.concurrent.locks.Condition
  - java.util.concurrent.Semaphore
- **For interprocess communication, see the later discussion on "Binder".**

Memory Management, Virtual Memory, Etc.

- ***ashmem* (Android Shared Memory)**
  - See later discussion on ashmem.
- **Applications use the Dalvik VM, and the memory management inherent in it.**
- **The Android Activity Manager manages system resources at a higher level**
  - Killing and restoring processes as needed.

File Systems and/or Persistent Storage

- **YAFFS2**
  - Yet Another Flash File System (2)
  - Optimized for flash memory
- **PMEM:**
  - Large (1-16+MB), physically contiguous regions of memory shared between user space and kernel drivers (dsp, gpu, etc).
  - Written specifically to deal with hardware limitations of MSM7201A, but could be used for other chipsets as well.

I/O and Graphics

- **Some drivers to support specific hardware**
  - No other major changes.

- **X Window system is *not* present**
  - Replaced by the *Surface Manager* and the *Window Manager*.

Program Loading & Unloading

- **Each application instance runs as its own process**
  - On top of its own VM!
- **A process is merely the current container for an application**
  - See Processes & Concurrency 

Networking and Communication

- **Added "paranoid networking" option – restricts some networking features, *depending upon the group ID of the calling process*.**

- **A lot more to be said**
  - For another course

Security

- Begins by relying on the Linux process-level independence. Each application is given its own UID and GID.
- Interaction between applications is via Binder, and is specified in the application's manifest.
- Permissions (in application manifest) are evaluated at installation-time, and either granted or denied based on checks with trusted authorities and user feedback.

Multiprocessor Support

- No changes from standard Linux kernel.

Linux Kernel Differences

- **Added: Generic, standalone Android-related drivers**
  - ashmem (Android shared memory implementation)
  - Low memory killer (resource management)
  - Binder: Interprocess communication (IPC)
  - Logger
- **Modifications: wakelocks**

We'll look at each these individually

- **Added: Hardware-specific additions**
  - **Support for new SoCs (System on a Chip)**
- **Added: Miscellaneous smaller changes – not significant from the perspective of system architecture & functionality**
- **Political Side-Note on Linux Kernel Kerfuffle: 1, 2**

ashmem

- **Android Shared Memory**
  - Fairly standard shared memory implementation,
  - Optimized for low-memory devices
  - Reference-counted objects – kernel can free shared memory no longer in use
  - Lives in virtual memory

Low Memory Killer

- **Used when memory is running low**
- **Selects a process to be killed, using hints from user-space**
- **Based on *Importance***
  - Kills lowest importance process
- **Foreground is the most important**
  - Process next back in the stack is likely to be more important than one deeper back in the stack
- **Used by the Activity Manager**

Binder

- **Originally from BeOS, CORBA-like IPC**
  - Much like Microsoft COM
- **Essentially, a *Remote Procedure Call* system**
  - Discussed in CS-4513, Distributed Systems
- **Purpose — app needs to call a function implemented by another app**
  - Using resources / data of that other app
- **…**

Binder (continued)

- …
- **Protocol for making a function call across process boundaries is easy**
  - Many standards, widely used
- **Finding what to call is hard!**
  - And where it is, etc.
- **Especially at run-time**
  - Even if you don't know what is installed
- **Many standards:–**
  - CORBA, Java RMI, DCE, Microsoft DCOM, Sun ONC, …

Intents — a protocol by which …

- **App declares "intention" to do something**
  - E.g., get directions to a particular restaurant
- **Other apps declare ability / willingness to cause that something to be done**
  - E.g., provide directions
- **Binding happens at run-time**
  - Without either app knowing what the other one is!
- **Lots of security and other issues**
  - Beyond the scope of this course

Summary

- Android is based on Linux, but is not Linux
- Many facilities for mobile platforms and applications
- Currently, the most widely used operating system on earth!

Questions?

Android: Summary

- Component-based architecture for mobile device development.

Questions? Etc.

- Any questions, discussion, etc.?

Background & Reference Materials

Resources Used

Burnett, Ed, “Hello, Android”, 3rd edition, 2010.

Meier, Reto, “Professional Android 2 Application Development”

Main Android Developer Site (both for application creators and for device manufacturers who want to run Android)

Android Modifications to Linux Kernel

Short Summary, Android Kernel Features

Porting Android: 1, 2

Security

http://developer.android.com/guide/topics/security/security.html

https://www.isecpartners.com/files/iSEC_Android_Exploratory_Blackhat_2009.pdf

http://siis.cse.psu.edu/slides/android-sec-tutorial.pdf