Android Development

What is Android?

- Android is a mobile operating system that is based on a modified version of Linux
- originally developed by a startup of the same name, Android, Inc
- In 2005 Google purchased Android
- Free and Open source software

Application Fundamentals

Android apps can be written using Kotlin, Java, and C++ languages. The Android SDK tools compile your code along with any data and resource files into an APK, an Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Each Android app lives in its own security sandbox, protected by the following Android security features:

The Android operating system is a multi-user Linux system in which each app is a different user.

By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.

Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.

By default, every app runs in its own Linux process. The Android system starts the process when any of the app's components need to be executed, and then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

The Android system implements the principle of least privilege.

It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files. To conserve system resources, apps with the same user ID can also arrange to run in the same Linux process and share the same VM. The apps must also be signed with the same certificate.

An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, and Bluetooth. The user has to explicitly grant these permissions.

App components

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

There are four different types of app components:

- Activities
- Services
- Broadcast receivers
- Content providers

Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. The following sections describe the four types of app components.

Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app
You implement an activity as a subclass of the Activity class.

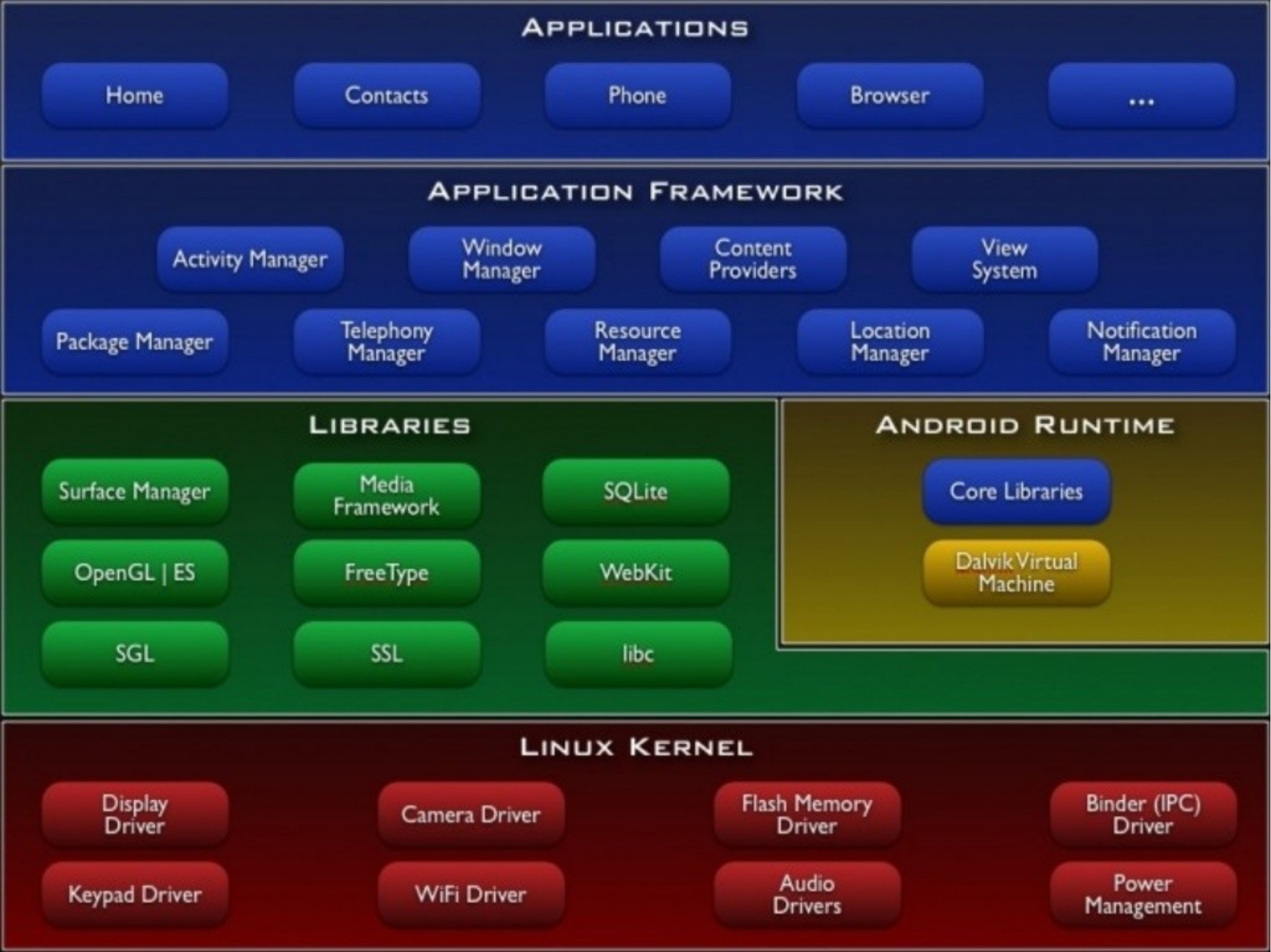
Activating components

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an intent. Intents bind individual components to each other at runtime. You can think of them as the messengers that request an action from other components, whether the component belongs to your app or another.

Android Architecture

The software stack is split into 5 layers

1) The application layer
2) The application framework
3) The android runtime
4) Libraries
5) Linux Kernel

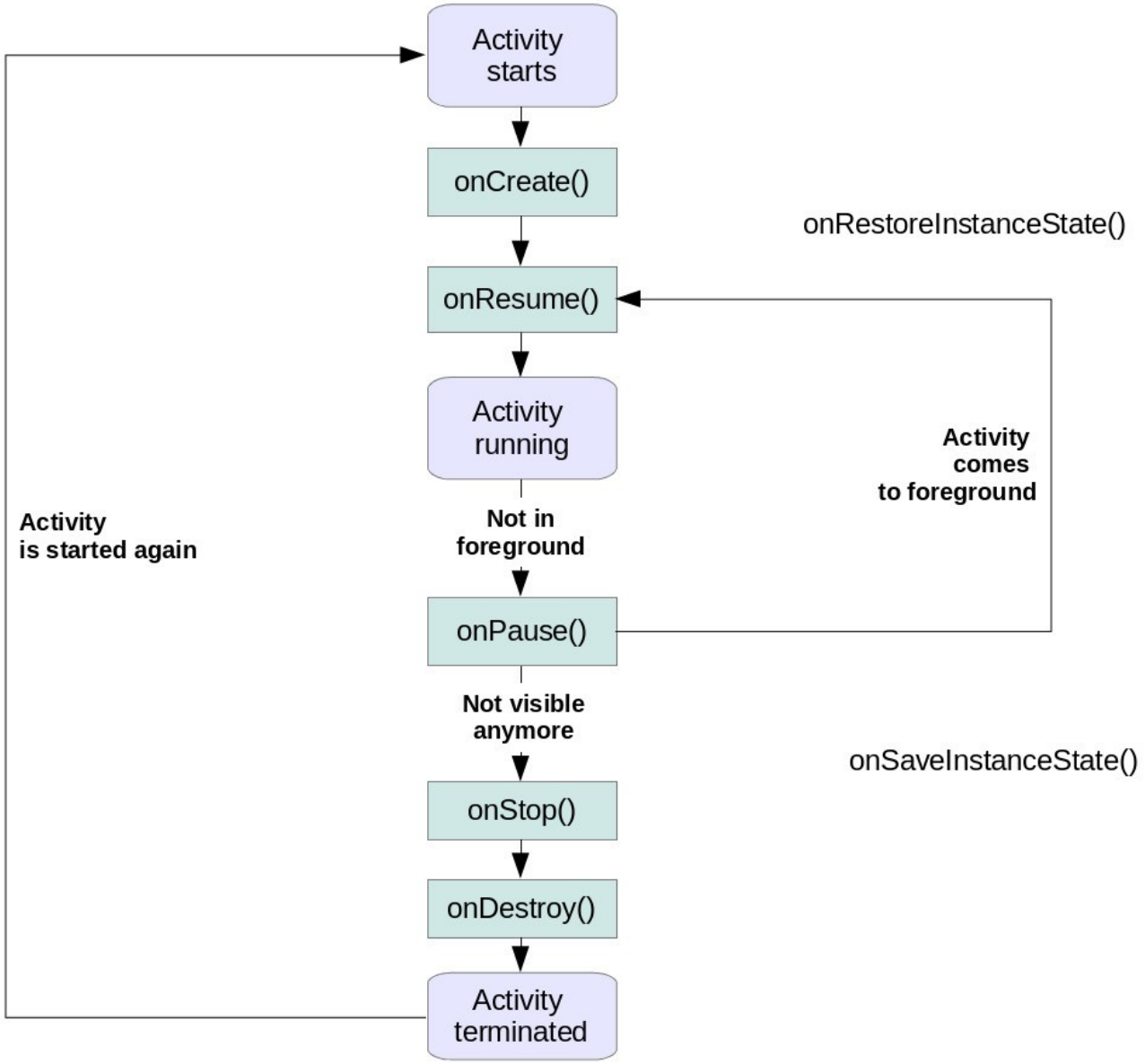

Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
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

Android Architecture

- Linux kernel
  - contains all the low-level device drivers for the various hardware components of an Android device
- Libraries
  - contain all the code that provides the main features of an Android OS
- Android Runtime
  - provides a set of core libraries that enable developers to write Android apps using the Java programming language
  - Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine
  - Android applications are compiled into the Dalvik executables

- Application Framework
  - Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications
  - This is all written in java. These applications include the ones that come with a phone or which you will be developing soon.
- Applications
  - Includes applications that ship with the Android device
  - as well as applications that you download and install from play store

Android Life Cycle



Application programming interface (API)

In computer programming, an application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication between various components.

(This is what Wikipedia says)

Just as a graphical user interface (GUI) makes it easier for people to use programs, application programming interfaces make it easier for developers to use certain technologies in building applications. By abstracting the underlying implementation and only exposing objects or actions the developer needs, an API simplifies programming.

For ex – Teleduino API.

Teleduino API

Teleduino converts your ethernet enabled Arduino into a powerful and versatile tool for interacting with devices over the internet. Not only that, but it makes it quick and easy.

Tasks you can perform

- Reset, ping, get uptime, get free memory.
- Define pin modes, set digital outputs, set analog outputs, read digital inputs, read analog inputs, or read all inputs with a single API call.
- Define up to 2 'banks' (4 for the Mega) of shift registers. Each 'bank' can contain up to 32 cascaded shift registers, giving a total of 512 digital outputs (1024 for the Mega).
- Shift register outputs can be set, or merged, and expire times can be set on merges (you could set an output(s) high for X number of milliseconds).
- Define, and read and write from serial port (4 for the Mega).
- Read and write from EEPROM.
- Define and position up to 6 servos (48 for the Mega).
- Interface with I2C (TWI) sensors and devices.
- Set preset values for the above functions, which get set during boot. Preset values are stored in the first 178 bytes of the EEPROM (413 for the Mega).