Dr. Sanjay Adiwal

Joint Director

Centre for Development of Advanced Computing (C-DAC)

Electronics City, Bangalore 560 100

Android

Android

- Software Stack for Mobile Devices that includes
  - OS
  - Middleware
  - Applications
- It is a Linux based platform for mobile devices and is an open source software.
- Application framework enabling reuse/replacement of Apps
- Dalvik VM optimized for mobile
- Optimized graphics
- SQLite for data storage
- Media Support
- Support for radio interfaces, Bluetooth, WIFI, camera
- First phone launched HTC GI in 2008

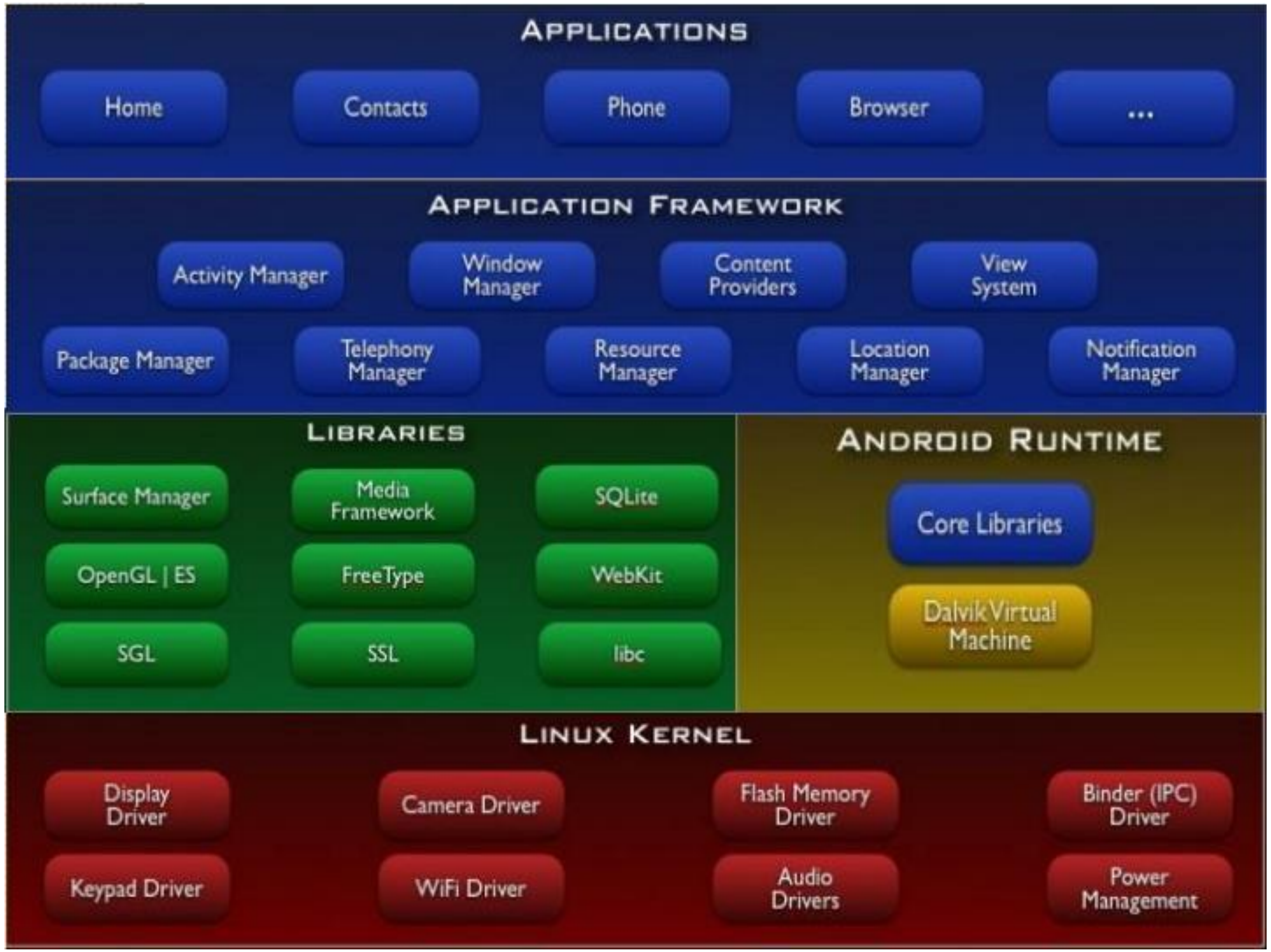
OPEN HANDSET ALLIANCE (OHA)

It's a consortium of several companies

CHINA MOBILE
ASUS
SoftBank
T-Mobile
AKM
Google
SiRF
Synaptics
KDDI
GARMIN
Sprint
TELECOM
esmertec
Audience
QUALCOMM
TELECA
NTT DoCoMo
htc smart mobility
MOTOROLA
TOSHIBA
Ascender Corporation
ebay
ARM
SkyPoP
HUAWEI
LG Electronics
SAMSUNG
Sony Ericsson
Telefonica
vodafone
NUANCE
pv
ERICSSON
BROADCOM
OMRON
ATHEROS
Living Image
noser
WIND RIVER
TEXAS INSTRUMENTS
BORQS
Aplix Corporation
SONiVOX sound that rocks
MARVELL
intel
NVIDIA
tat

- A business alliance consisting of 47 companies to develop open standards for mobile devices

Android Architecture



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

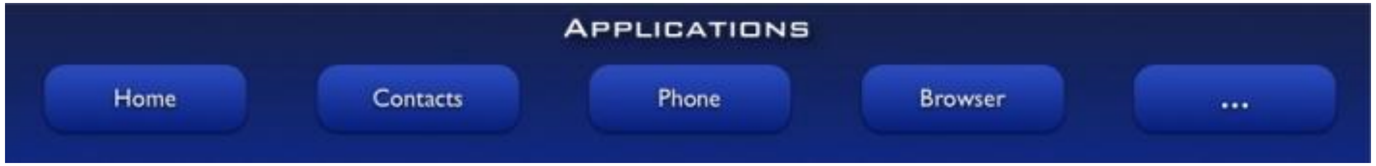
Display Driver, Camera Driver, Flash Memory Driver, Binder (IPC) Driver

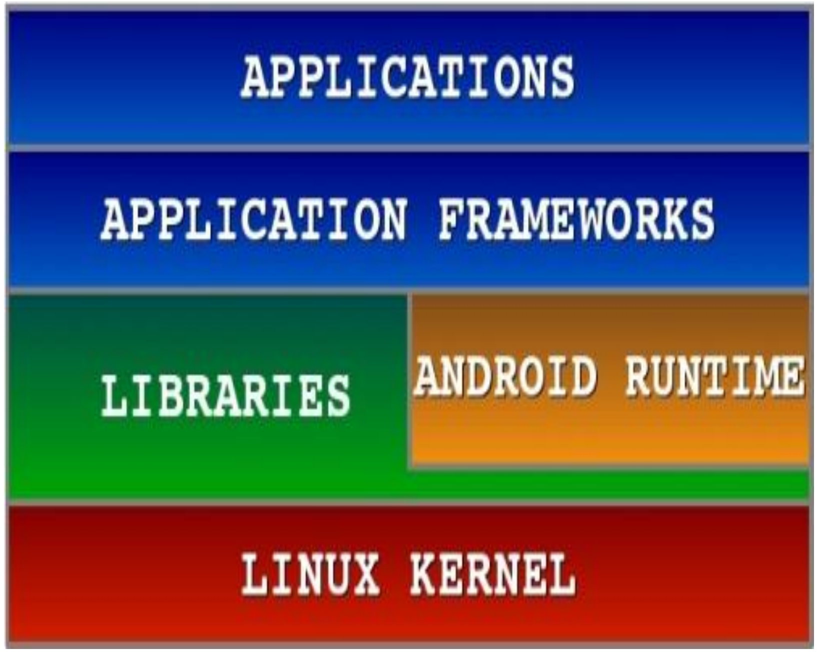
Keypad Driver, WiFi Driver, Audio Drivers, Power Management

Applications

- Android provides a set of core applications:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc

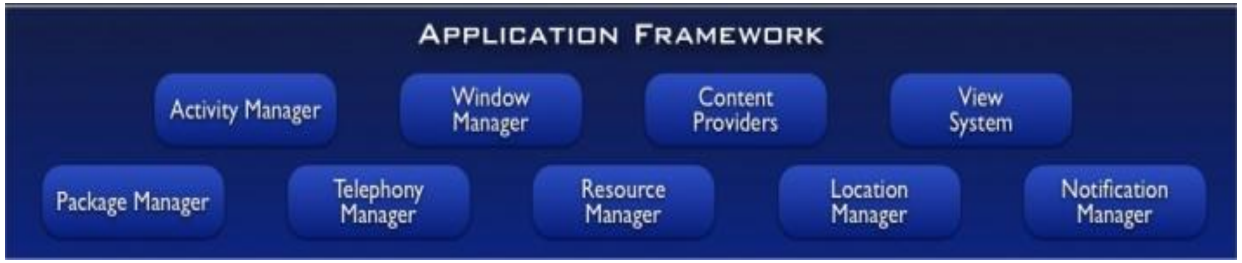
- All applications are written using the Java language.

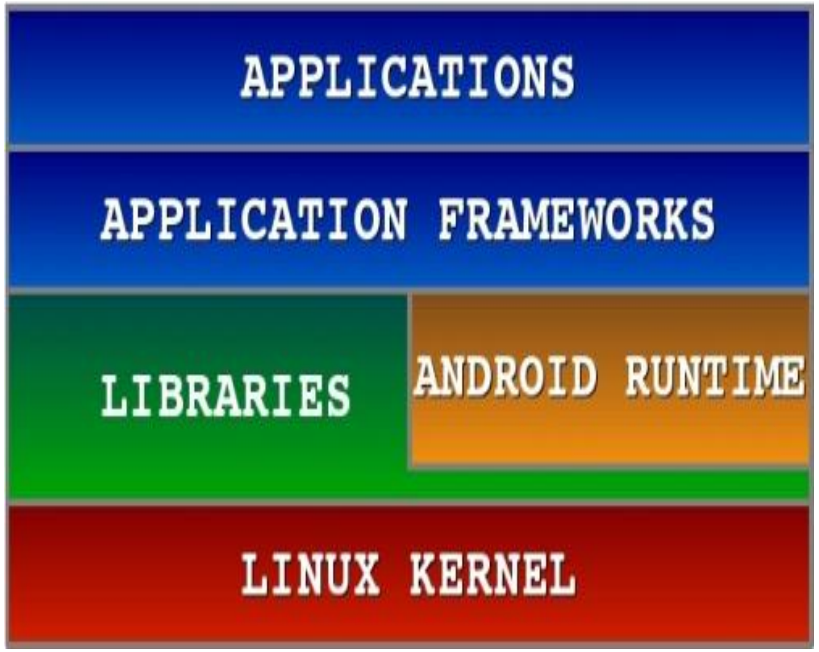




Application Framework

- Enabling and simplifying the reuse of components
  - ✓ Developers have full access to the same framework APIs used by the core applications.
  - ✓ Users are allowed to replace components.



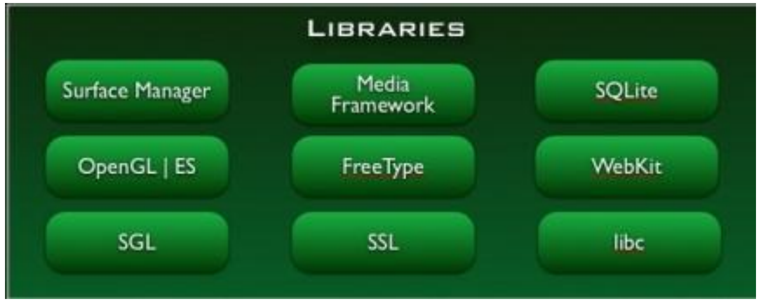


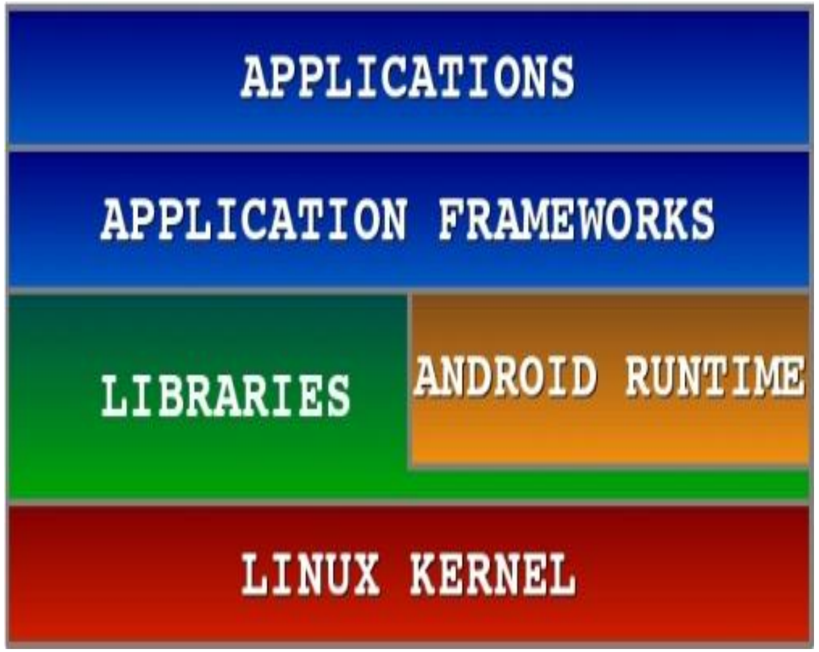
App Framework

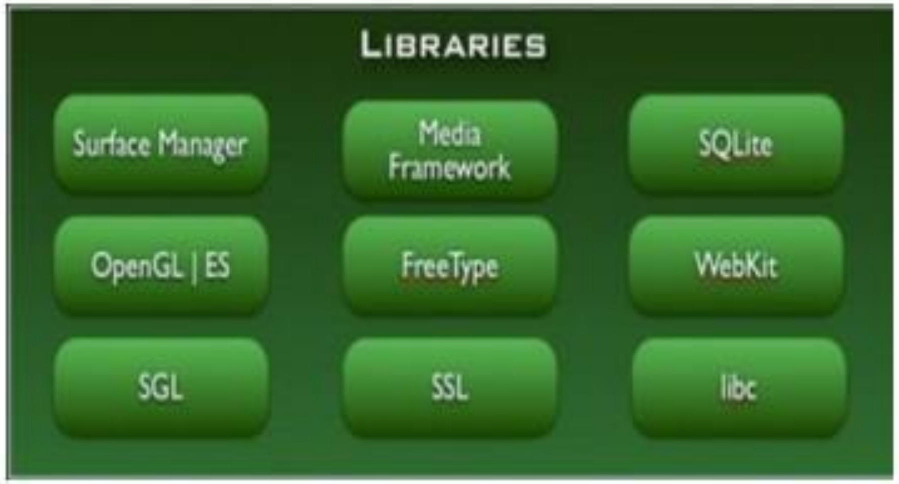
Feature	Role
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized strings, graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

Libraries

- Including a set of C/C++ libraries used by components of the Android system
- Exposed to developers through the Android application framework







- Libc:c standard library
- SSL: Secure Socket Layer

- Surface Manager: responsible for composing different drawing surfaces onto the screen.
- OpenGL|ES : 3D Image Engine
- SGL : 2D image Engine.

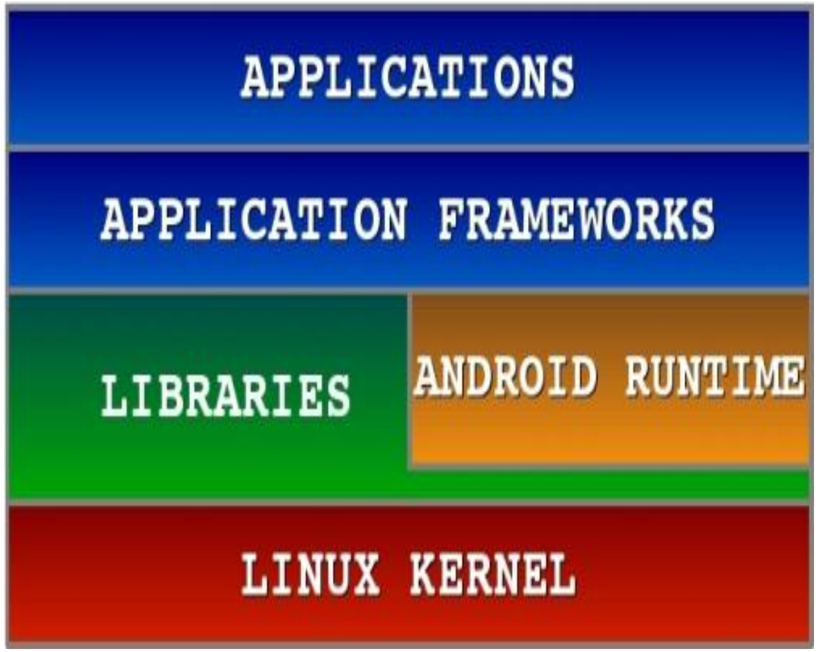
➔ Hence we can combine 3D and 2D graphics in the same application.

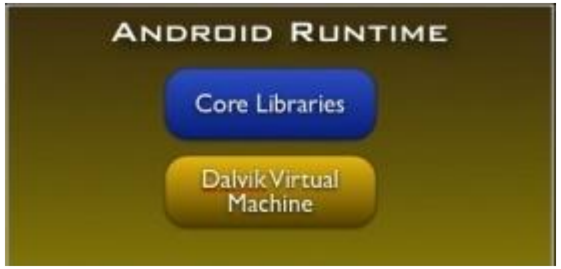
- Media Framework : Core part of the android multimedia.➔ MPEG4,H264,MP3,AAC…..

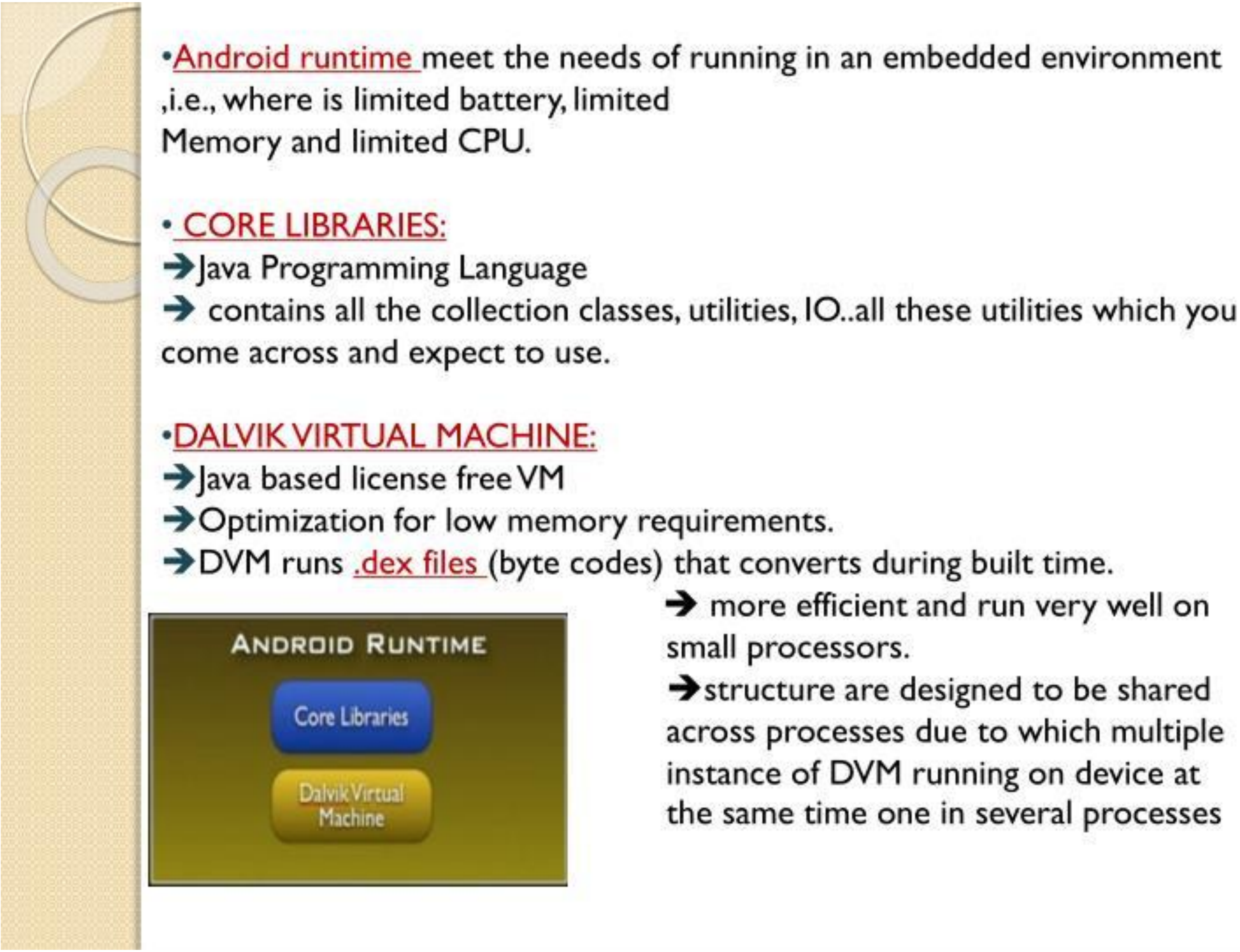
- FreeType: To render the fonts.
- WebKit:open source browser engine. Helps to work well on small screen.
- SQLite: Embedded Database

Android Runtime

- Core Libraries
  - ✓Providing most of the functionality available in the core libraries of the Java language
  - ✓APIs
    - ➢Data Structures
    - ➢Utilities
    - ➢File Access
    - ➢Network Access
    - ➢Graphics
    - ➢Etc







•Android runtime meet the needs of running in an embedded environment ,i.e., where is limited battery, limited
Memory and limited CPU.

• CORE LIBRARIES:

➔Java Programming Language

➔ contains all the collection classes, utilities, IO..all these utilities which you come across and expect to use.

•DALVIK VIRTUAL MACHINE:

➔Java based license free VM

➔Optimization for low memory requirements.

➔DVM runs .dex files (byte codes) that converts during built time.

➔ more efficient and run very well on small processors.

➔structure are designed to be shared across processes due to which multiple instance of DVM running on device at the same time one in several processes

ANDROID RUNTIME

Core Libraries

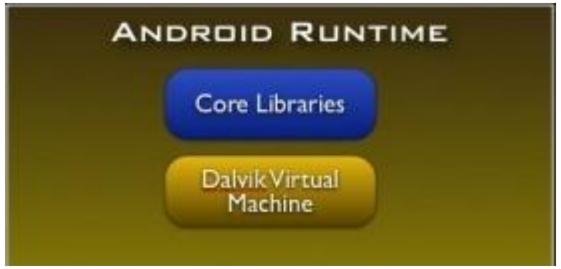
Dalvik Virtual Machine

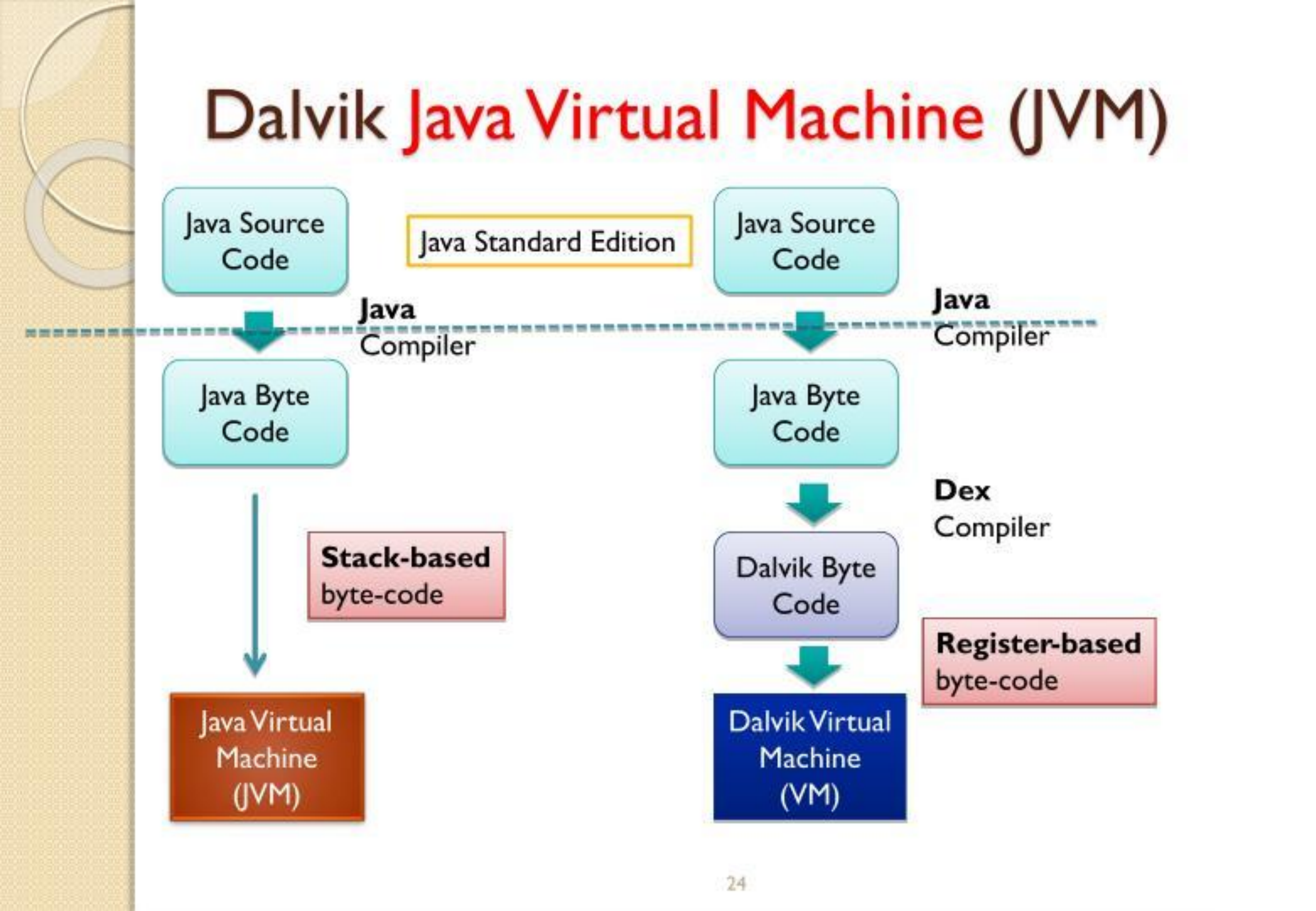
Dalvik Virtual Machine

- Providing environment on which every Android application runs
    - ➢ Each Android application runs in its own process, with its own instance of the Dalvik VM.
    - ➢ Dalvik has been written such that a device can run multiple VMs efficiently.

  Register-based virtual machine

- All the applications written in JAVA are convered to the dalvik executables .dex.
- Every android app runs its own process, with its own instances of the DVM.





Dalvik Java Virtual Machine (JVM)

Java Standard Edition

Java Source Code

Java Compiler

Java Byte Code

Stack-based byte-code

Java Virtual Machine (JVM)

Java Source Code

Java Compiler

Java Byte Code

Dex Compiler

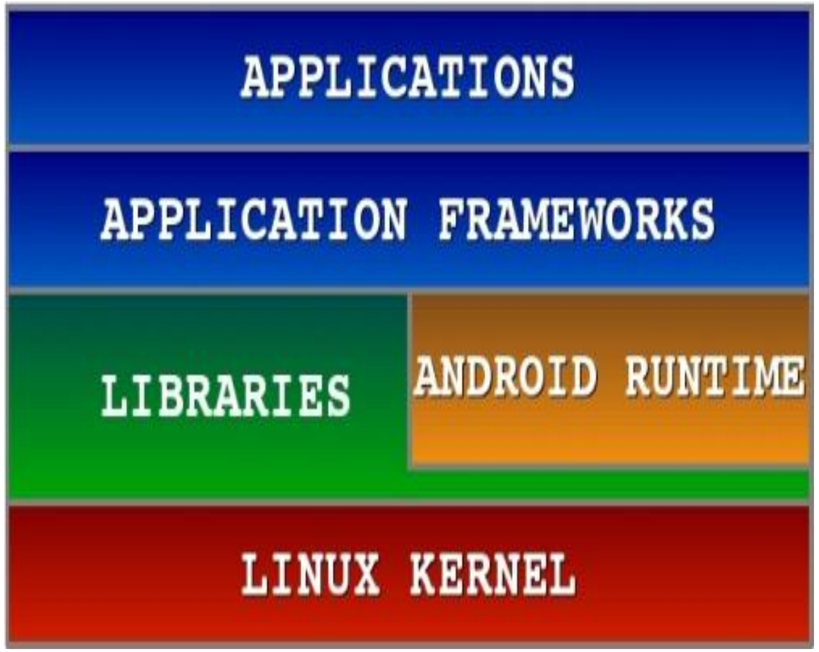
Dalvik Byte Code

Register-based byte-code

Dalvik Virtual Machine (VM)

Linux Kernel

- Relying on Linux Kernel for core system services
  - ✓ Memory and Process Management
  - ✓ Network Stack
  - ✓ Driver Model
  - ✓ Security
- Providing an abstraction layer between the H/W and the rest of the S/W stack



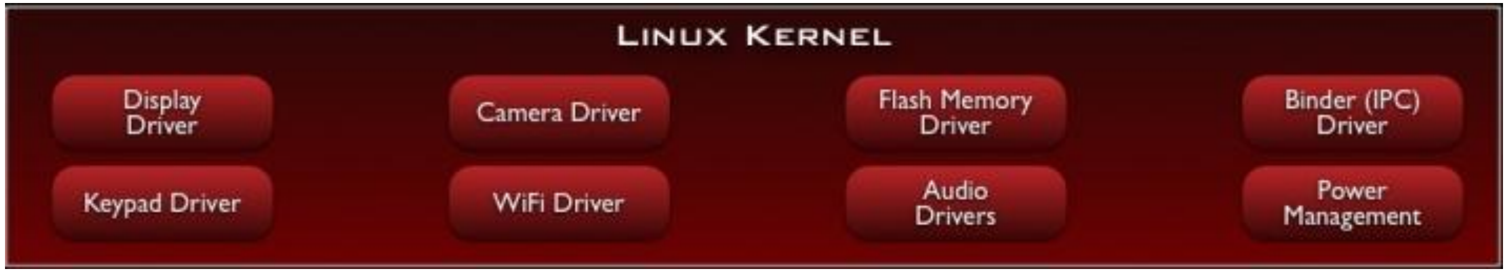
APPLICATIONS

APPLICATION FRAMEWORKS

LIBRARIES

ANDROID RUNTIME

LINUX KERNEL



LINUX KERNEL

Display Driver

Camera Driver

Flash Memory Driver

Binder (IPC) Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management

Root a Phone

- Rooting refers to obtaining access to commands, system files, and folder locations that are usually locked off.
- Rooting Android can be thought of as promoting yourself from a system user to an administrator, with the additional freedom and risks that come from more control over the deeper workings of your device.
- adding root to Android means becoming a superuser.
- it gives you more access to several cool apps and customization options
- It also lets you upgrade older phones to newer Android versions long after your manufacturer and carrier have stopped supporting it.

Rooting Terminologies

- Bootloader: Lowest level software on your phone that starts up recoveries and then the main operating system.
- Recovery: Low-level software that can create and restore full system backups. It is accessed before the main OS.
- ADB (Android Debug Bridge): A command-line tool that is part of the Android SDK, which supports communication between a computer and an Android device.

Android root apps and options

- Overclock or underclock the CPU.
- Increase battery life.
- Greatly enhance the power of Tasker
- Remove preinstalled bloatware apps.
- Make real backups.
- Install custom ROMs.
- Further customize your phone’s looks.
- Install apps that do more than basic ones.
- You can even do things like installing Ubuntu for desktop.

How to root Android devices

- Different brands and even software versions can make the rooting process vary.
- Even within handset variants, you may find that some techniques work and others don’t.
- Easiest method is usually to use a simple root app
- These apps let you root Android with a single tap, with some popular examples being KingRoot, KingoRoot, and OneClickRoot.
- Before rooting your device, it’s always good to back up first.

What is ADB?

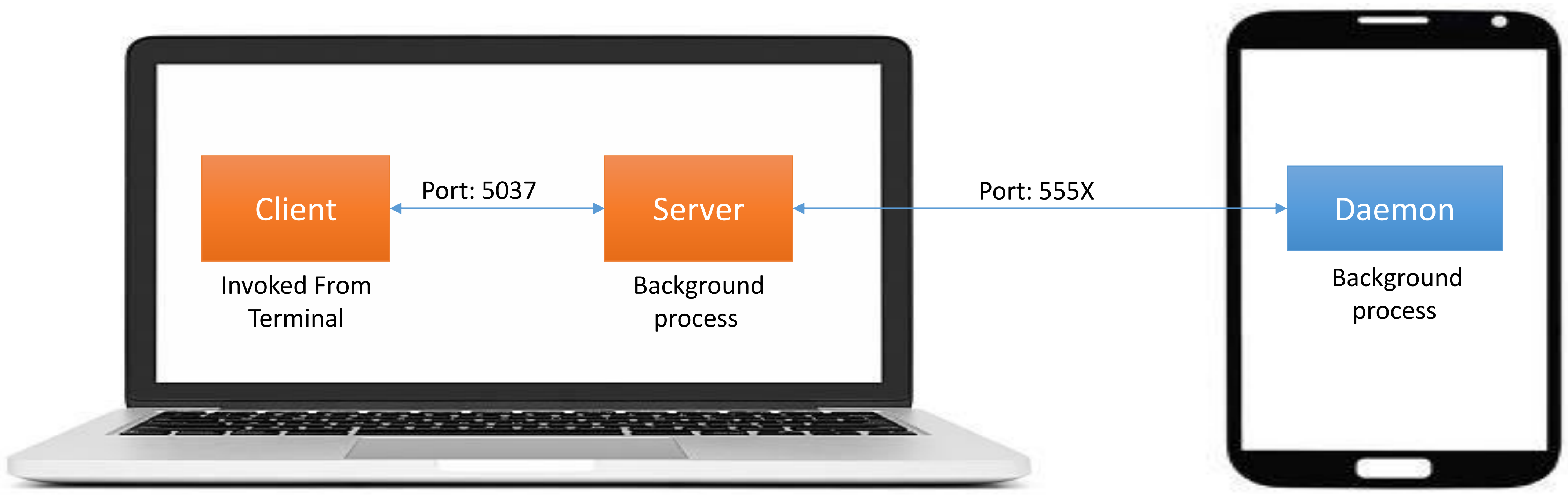
- Android Debug Bridge (ADB) is a **command line utility** to communicate with your Android device.

https://developer.android.com/studio/command-line/adb

- **Uses:**
  - Listing connected devices.
  - Installing and debugging apps.
  - Copy files to and from the phone.
  - Take screenshots, record screens, etc.

ADB Client-Server Architecture

- Three entities are involved: a client, a server, a daemon (adbd).

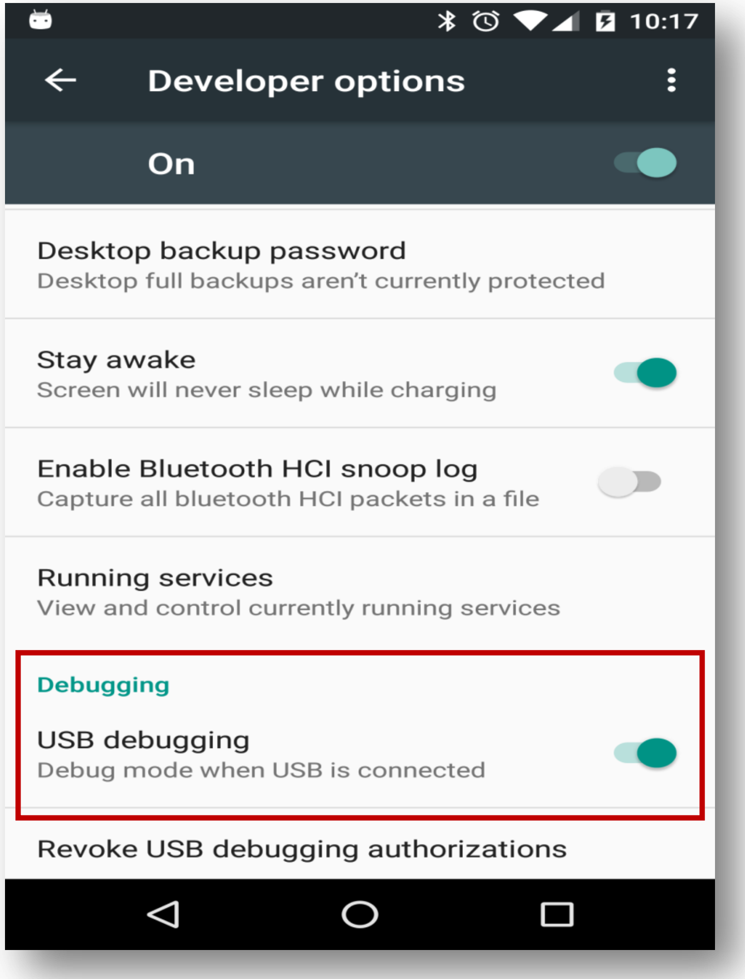


Connecting over USB

- Remember the first thing we did with the phone?

```
Settings>About Phone> Build number
```

```
Settings>Developer options> USB Debugging
```



10:17

Developer options

On

Desktop backup password
Desktop full backups aren't currently protected

Stay awake
Screen will never sleep while charging

Enable Bluetooth HCI snoop log
Capture all bluetooth HCI packets in a file

Running services
View and control currently running services

Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorizations

Testing the ADB

- Use a shell to invoke your first command:

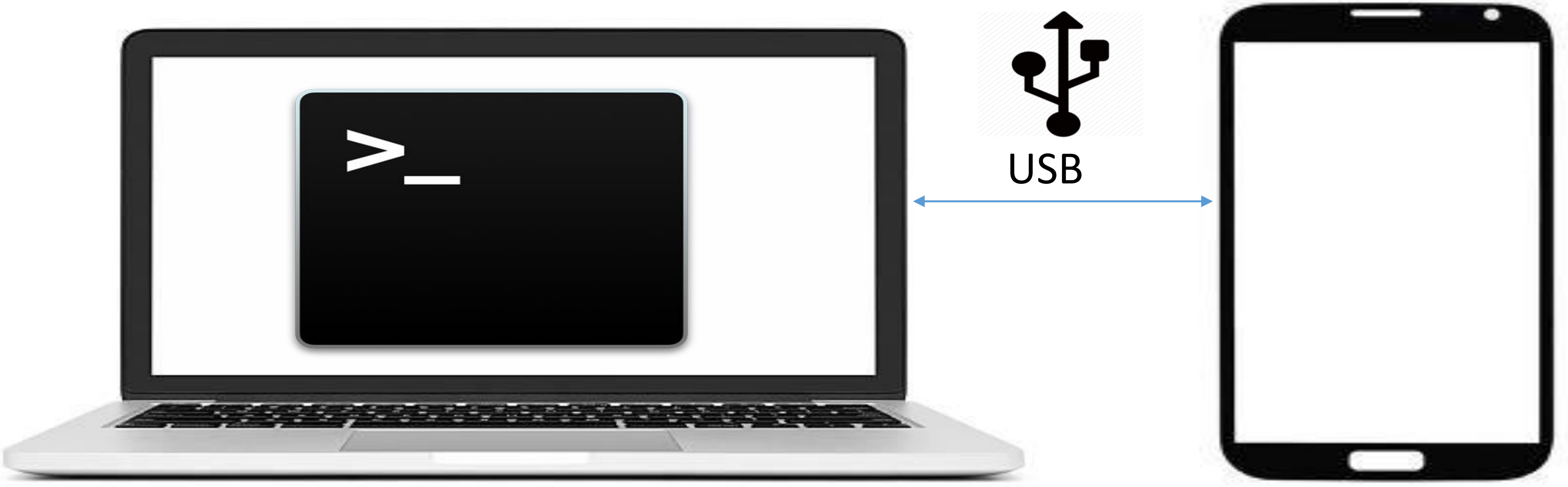
```
➢ adb devices -l

List of devices attached
071b799a344bdc6b       device product:hammerhead model:Nexus_5 device:hammerhead
```

When things go wrong …

- Use a shell to kill and start the server:

```
➢ adb kill-server
➢ adb start-server
* daemon not running. starting it now at tcp:5037 *
* daemon started successfully *
```



ADB over WiFi

- Set TCP port (over USB) and connect (over WiFi)

```
➢ adb tcpip 5555
```

(USB must be connected)

```
➢ adb connect [Phone's IP Addr]
```

(USB must be disconnected)

To get your phone's IP address: `adb shell ifconfig`

ADB Commands

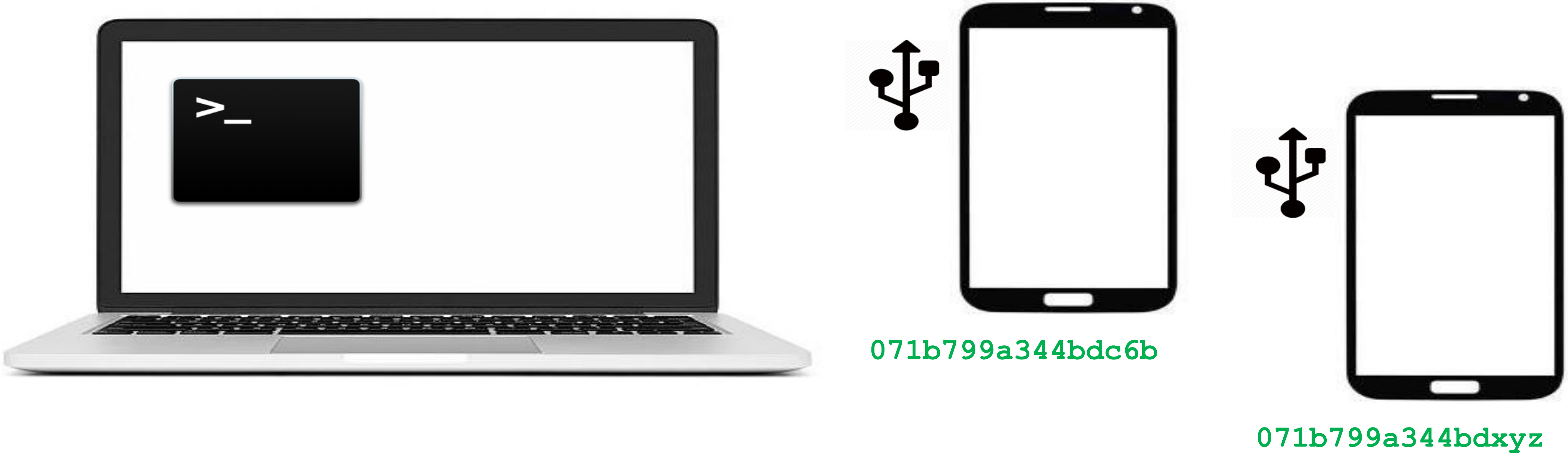
- Now, you can communicate with the phone.

```
➢ adb shell screencap /sdcard/s.png
➢ adb pull /sdcard/s.png <local>
```

Useful ADB Commands

- **Global Options:**

```
adb -d <command>
adb -e <command>
adb -s <serial number> <command>
```



Useful ADB Commands

- **General adb Commands:**

```
➢ adb devices -l
➢ adb help
➢ adb version
```

- **Network/File/Install adb Commands:**

```
➢ adb connect ip[:port]
➢ adb disconnect ip[:port]
```

```
➢ adb pull [-a] remote local
➢ adb push local remote
```

```
➢ adb install package
➢ adb uninstall package
```

Useful ADB Commands

- **backup/debug/scripting adb Commands:**

```
➢ adb backup [-f file] [-apk][-obb][-shared][-all][-system]
➢ adb restore file
```

```
➢ adb logcat [option]
```

```
➢ adb root
➢ adb unroot
➢ adb usb
➢ adb tcpip port_555X
```

```
➢ adb start-server
➢ adb kill-server
➢ adb reconnect
```

Issuing shell Commands

- Generic format (two ways):

```
➢ adb shell <command>           (execute on device shell)
➢ adb shell                       (get a device shell)
```

- Activity Manager (am commands)

```
➢ am start [option] intent (e.g., -a android.media.action.VIDEO_CAMERA)
➢ am startservice [option] intent
➢ am broadcast [option] intent
➢ am force-stop package
➢ am kill [option] package
➢ am kill-all
```

Issuing shell Commands

- Package Manager (pm commands)

```
➢ pm list packages [options] filter
➢ pm list permissions [options] filter
➢ pm list features
➢ pm list libraries
➢ pm list users
➢ pm install [options] path
➢ pm uninstall [options] package
➢ pm grant package permission
➢ pm revoke package permission
```

- Screen capture

```
➢ screencap filename
➢ screenrecord [option] filename
```

Issuing shell Commands

- **Many more Unix commands!**

```
➢ echo text
➢ date
➢ time
➢ reboot
```

```
➢ ifconfig
➢ iftop
➢ netstat
➢ ping host
```

```
➢ ls
➢ cd directory
➢ chmod
➢ df
➢ mkdir
➢ rm file
➢ rmdir dir
```

```
➢ ps
➢ top
```

Shell commands from Android APPs

- Use **‘Runtime’** and **‘Process’** classes to issue shell commands from an Android app:

```
try {
    Process p = Runtime.getRuntime().exec("ps");

    InputStreamReader isr = new                 InputStreamReader(p.getInputStream());
    BufferedReader br = new BufferedReader(isr);
    String str ="";

    while((str = br.readLine()) != null){
        Log.v("Tag", str);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Thank You