**Android** is a [Linux](https://en.wikipedia.org/wiki/Linux)-based [operating system for mobile devices](https://en.wikipedia.org/wiki/Mobile_operating_system) such as [smartphones](https://en.wikipedia.org/wiki/Smartphone) and [tablet computers](https://en.wikipedia.org/wiki/Tablet_computer). It is developed by the [Open Handset Alliance](https://en.wikipedia.org/wiki/Open_Handset_Alliance), led by [Google](https://en.wikipedia.org/wiki/Google).  Google releases the Android code as [open-source](https://en.wikipedia.org/wiki/Open-source), under the [Apache License](https://en.wikipedia.org/wiki/Apache_License).[[10]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-AndroidOverview-9) The [Android Open Source Project](https://en.wikipedia.org/wiki/Android_Open_Source_Project) (AOSP) is tasked with the maintenance and further development of Android.[[11]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-source.android.com-10)

Android has been updated frequently since the original release of "Astro", with each fixing [bugs](https://en.wikipedia.org/wiki/Software_bug) and adding new features. Each version is named in alphabetical order, with 1.5 "Cupcake" being the first named after a [dessert](https://en.wikipedia.org/wiki/Dessert) and every update since following this naming convention.[[35]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-34)

List of Android version names:

* Cupcake
* Donut
* Eclair
* Froyo
* Gingerbread
* Honeycomb
* Ice Cream Sandwich
* Jelly Bean

Android consists of a [kernel](https://en.wikipedia.org/wiki/Kernel_(software)) based on the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel), with [middleware](https://en.wikipedia.org/wiki/Middleware), [libraries](https://en.wikipedia.org/wiki/Software_library) and [APIs](https://en.wikipedia.org/wiki/Application_programming_interface) written in [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [application software](https://en.wikipedia.org/wiki/Application_software) running on an[application framework](https://en.wikipedia.org/wiki/Application_framework) which includes Java-compatible libraries based on [Apache Harmony](https://en.wikipedia.org/wiki/Apache_Harmony). Android uses the [Dalvik virtual machine](https://en.wikipedia.org/wiki/Dalvik_(software)" \o "Dalvik (software)) with [just-in-time compilation](https://en.wikipedia.org/wiki/Just-in-time_compilation) to run Dalvik dex-code (Dalvik Executable), which is usually translated from [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) bytecode.[[36]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-35)

The main hardware platform for Android is the [ARM architecture](https://en.wikipedia.org/wiki/ARM_architecture).

**ARM** is a [reduced instruction set computer](https://en.wikipedia.org/wiki/Reduced_instruction_set_computer) (RISC) [instruction set architecture](https://en.wikipedia.org/wiki/Instruction_set_architecture) (ISA) developed by [ARM Holdings](https://en.wikipedia.org/wiki/ARM_Holdings). It was named the *Advanced RISC Machine* and, before that, the *Acorn RISC Machine*.\

**Apache Harmony** was an [open source](https://en.wikipedia.org/wiki/Open_source), [free Java implementation](https://en.wikipedia.org/wiki/Free_Java_implementations), developed by the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation).

[Dalvik](https://en.wikipedia.org/wiki/Dalvik_virtual_machine), the virtual machine used in [Google](https://en.wikipedia.org/wiki/Google)'s [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) platform, uses a subset of Harmony for the core of its [Class Library](https://en.wikipedia.org/wiki/Java_Class_Library).[[15]](https://en.wikipedia.org/wiki/Apache_Harmony#cite_note-14) However, Dalvik does not align to [Java SE](https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition) nor [Java ME](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition) [Class Library](https://en.wikipedia.org/wiki/Java_Class_Library)profiles (for example [J2ME](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition) classes, [AWT](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit) and [Swing](https://en.wikipedia.org/wiki/Swing_(Java)) are not supported). Instead it uses its own library,[[16]](https://en.wikipedia.org/wiki/Apache_Harmony" \l "cite_note-15) built on a subset of Harmony.

**Features of the Android™ kernel**

Current features and specifications:[[49]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)" \l "cite_note-WhatIsAndroid-48)[[50]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-EnSDK-49)[[51]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-mediaformats-50)

**Handset layouts**

The platform is adaptable to larger, [VGA](https://en.wikipedia.org/wiki/Video_Graphics_Array), [2D graphics](https://en.wikipedia.org/wiki/2D_computer_graphics) library, [3D graphics](https://en.wikipedia.org/wiki/3D_computer_graphics) library based on [OpenGL ES](https://en.wikipedia.org/wiki/OpenGL_ES) 2.0 specifications, and traditional smartphone layouts.

**Storage**

[SQLite](https://en.wikipedia.org/wiki/SQLite), a lightweight [relational database](https://en.wikipedia.org/wiki/Relational_database), is used for [data](https://en.wikipedia.org/wiki/Data) storage purposes.

**Connectivity**

Android supports connectivity technologies including [GSM](https://en.wikipedia.org/wiki/GSM)/[EDGE](https://en.wikipedia.org/wiki/Enhanced_Data_Rates_for_GSM_Evolution), [IDEN](https://en.wikipedia.org/wiki/Integrated_Digital_Enhanced_Network), [CDMA](https://en.wikipedia.org/wiki/Code_division_multiple_access), [EV-DO](https://en.wikipedia.org/wiki/Evolution-Data_Optimized), [UMTS](https://en.wikipedia.org/wiki/Universal_Mobile_Telecommunications_System), [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth), [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi), [LTE](https://en.wikipedia.org/wiki/LTE_Advanced), [NFC](https://en.wikipedia.org/wiki/Near_field_communication) and [WiMAX](https://en.wikipedia.org/wiki/WiMAX).

**Messaging**

[SMS](https://en.wikipedia.org/wiki/SMS) and [MMS](https://en.wikipedia.org/wiki/Multimedia_Messaging_Service) are available forms of messaging, including threaded [text messaging](https://en.wikipedia.org/wiki/Text_messaging) and [Android Cloud To Device Messaging](https://en.wikipedia.org/wiki/Android_Cloud_To_Device_Messaging) (C2DM) and now enhanced version of C2DM, Android [Google Cloud Messaging](https://en.wikipedia.org/wiki/Google_Cloud_Messaging) (GCM) is also a part of Android Push Messaging service.

**Multiple language support**

Android supports multiple languages.[[52]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-gingerbread-highlights-51)

**Web browser**

The web browser available in Android is based on the open-source [WebKit](https://en.wikipedia.org/wiki/WebKit" \o "WebKit) layout engine, coupled with [Chrome](https://en.wikipedia.org/wiki/Google_Chrome)'s [V8 JavaScript engine](https://en.wikipedia.org/wiki/V8_JavaScript_engine). The browser scores 100/100 on the [Acid3](https://en.wikipedia.org/wiki/Acid3#Mobile_browsers) test on Android 4.0.

**Java support**

While most Android applications are written in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), there is no [Java Virtual Machine](https://en.wikipedia.org/wiki/Java_Virtual_Machine) in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on [Dalvik](https://en.wikipedia.org/wiki/Dalvik_virtual_machine" \o "Dalvik virtual machine), a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. [J2ME](https://en.wikipedia.org/wiki/J2ME) support can be provided via third-party applications.

**Media support**

Android supports the following audio/video/still media formats: [WebM](https://en.wikipedia.org/wiki/WebM" \o "WebM), [H.263](https://en.wikipedia.org/wiki/H.263), [H.264](https://en.wikipedia.org/wiki/H.264) (in [3GP](https://en.wikipedia.org/wiki/3GP) or [MP4](https://en.wikipedia.org/wiki/MP4) [container](https://en.wikipedia.org/wiki/Container_format_(digital))), [MPEG-4 SP](https://en.wikipedia.org/wiki/MPEG-4_Part_2), [AMR](https://en.wikipedia.org/wiki/Adaptive_multi-rate_compression), [AMR-WB](https://en.wikipedia.org/wiki/AMR-WB) (in 3GP container), [AAC](https://en.wikipedia.org/wiki/Advanced_Audio_Coding), [HE-AAC](https://en.wikipedia.org/wiki/HE-AAC) (in MP4 or 3GP container), [MP3](https://en.wikipedia.org/wiki/MP3), [MIDI](https://en.wikipedia.org/wiki/Musical_Instrument_Digital_Interface), [Ogg Vorbis](https://en.wikipedia.org/wiki/Vorbis" \o "Vorbis), [FLAC](https://en.wikipedia.org/wiki/Free_Lossless_Audio_Codec), [WAV](https://en.wikipedia.org/wiki/WAV), [JPEG](https://en.wikipedia.org/wiki/JPEG), [PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics), [GIF](https://en.wikipedia.org/wiki/Graphics_Interchange_Format), [BMP](https://en.wikipedia.org/wiki/BMP_file_format), [WebP](https://en.wikipedia.org/wiki/WebP" \o "WebP).[[51]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-mediaformats-50)

**Streaming media support**

RTP/RTSP streaming ([3GPP PSS](https://en.wikipedia.org/w/index.php?title=3GPP_PSS&action=edit&redlink=1), [ISMA](https://en.wikipedia.org/wiki/Internet_Streaming_Media_Alliance)), HTML progressive download ([HTML5 <video> tag](https://en.wikipedia.org/wiki/HTML5_video)). Adobe Flash Streaming (RTMP) and HTTP Dynamic Streaming are supported by the [Flash plugin](https://en.wikipedia.org/wiki/Adobe_Flash_Player#Mobile_platforms).[[53]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-52) Apple HTTP Live Streaming is supported by [RealPlayer for Android](https://en.wikipedia.org/wiki/RealPlayer_for_Android),[[54]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)" \l "cite_note-53) and by the operating system in Android 3.0 (Honeycomb).[[55]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-honeycomb-highlights-54)

**Additional hardware support**

Android can use video/still cameras, [touchscreens](https://en.wikipedia.org/wiki/Touchscreen), [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System), [accelerometers](https://en.wikipedia.org/wiki/Accelerometer), [gyroscopes](https://en.wikipedia.org/wiki/Gyroscope), [barometers](https://en.wikipedia.org/wiki/Barometer), [magnetometers](https://en.wikipedia.org/wiki/Magnetometer), dedicated gaming controls, [proximity](https://en.wikipedia.org/wiki/Proximity_sensor) and [pressure sensors](https://en.wikipedia.org/wiki/Pressure_sensor),[thermometers](https://en.wikipedia.org/wiki/Thermometer), accelerated 2D [bit blits](https://en.wikipedia.org/wiki/Bit_blit) (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

**Multi-touch**

Android has native support for [multi-touch](https://en.wikipedia.org/wiki/Multi-touch) which was initially made available in handsets such as the [HTC Hero](https://en.wikipedia.org/wiki/HTC_Hero). The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time).[[56]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-55) Google has since released an update for the [Nexus One](https://en.wikipedia.org/wiki/Nexus_One) and the [Motorola Droid](https://en.wikipedia.org/wiki/Motorola_Droid) which enables multi-touch natively.[[57]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-56)

**Bluetooth**

Supports [A2DP](https://en.wikipedia.org/wiki/A2DP), [AVRCP](https://en.wikipedia.org/wiki/AVRCP), sending files ([OPP](https://en.wikipedia.org/wiki/Object_Push_Profile)), accessing the phone book ([PBAP](https://en.wikipedia.org/wiki/Bluetooth_profile#Phone_Book_Access_Profile_.28PBAP.2C_PBA.29)), voice dialing and sending contacts between phones. Keyboard, mouse and joystick ([HID](https://en.wikipedia.org/wiki/Bluetooth_profile#Human_Interface_Device_Profile_.28HID.29)) support is available in Android 3.1+, and in earlier versions through manufacturer customizations and third-party applications.[[58]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-57)

**Video calling**

Android does not support native video calling, but some handsets have a customized version of the operating system that supports it, either via the [UMTS](https://en.wikipedia.org/wiki/UMTS) network (like the [Samsung Galaxy S](https://en.wikipedia.org/wiki/Samsung_Galaxy_S)) or over IP. Video calling through Google Talk is available in Android 2.3.4 and later. Gingerbread allows [Nexus S](https://en.wikipedia.org/wiki/Nexus_S) to place Internet calls with a SIP account. This allows for enhanced VoIP dialing to other SIP accounts and even phone numbers. Skype 2.1 offers video calling in Android 2.3, including front camera support.

**Multitasking**

Multitasking of applications, with unique handling of memory allocation, is available.[[59]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-58)

**Voice based features**

Google search through voice has been available since initial release.[[60]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-59) Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards.[[61]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-60)

**Tethering**

Android supports [tethering](https://en.wikipedia.org/wiki/Tethering), which allows a phone to be used as a wireless/wired [Wi-Fi hotspot](https://en.wikipedia.org/wiki/Wi-Fi_hotspot). Before Android 2.2 this was supported by third-party applications or manufacturer customizations.[[62]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-61)

**Screen capture**

Android supports capturing a [screenshot](https://en.wikipedia.org/wiki/Screenshot) by pressing the power and volume-down buttons at the same time.[[63]](https://en.wikipedia.org/wiki/Android_(mobile_device_platform)#cite_note-62) Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android.

**External storage**

Most Android devices include microSD slot and can read microSD cards formatted with [FAT32](https://en.wikipedia.org/wiki/FAT32), [Ext3](https://en.wikipedia.org/wiki/Ext3) or [Ext4](https://en.wikipedia.org/wiki/Ext4) file system. To allow use of high-capacity storage media such as [USB flash drives](https://en.wikipedia.org/wiki/USB_flash_drive)and [USB HDDs](https://en.wikipedia.org/wiki/USB_HDD), many Android tablets also include [USB](https://en.wikipedia.org/wiki/USB) 'A' receptacle. Storage formatted with [FAT32](https://en.wikipedia.org/wiki/FAT32) is handled by [Linux Kernel](https://en.wikipedia.org/wiki/Linux_Kernel) VFAT driver, while 3rd party solutions are required to handle other popular file systems such as [NTFS](https://en.wikipedia.org/wiki/NTFS), [HFS Plus](https://en.wikipedia.org/wiki/HFS%2B) and [exFAT](https://en.wikipedia.org/wiki/ExFAT" \o "ExFAT).

**Android Application Development**

Applications are usually developed in the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) language using the [Android Software Development Kit](https://en.wikipedia.org/wiki/Android_SDK), but other development tools are available, including a [Native Development Kit](https://en.wikipedia.org/wiki/Android_NDK) for applications or extensions in C or C++, [Google App Inventor](https://en.wikipedia.org/wiki/Google_App_Inventor), a visual environment for novice programmers and various [cross platform mobile web applications frameworks](https://en.wikipedia.org/wiki/Multiple_phone_web_based_application_framework).

**App Inventor for Android** is an application originally provided by [Google](https://en.wikipedia.org/wiki/Google), and now maintained by the [Massachusetts Institute of Technology](https://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology) (MIT).

It allows anyone, (including people unfamiliar with [computer programming](https://en.wikipedia.org/wiki/Computer_programming)), to create [software applications](https://en.wikipedia.org/wiki/Application_software) for the [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) operating system (OS). It uses a graphical interface, very similar to [Scratch](https://en.wikipedia.org/wiki/Scratch_(programming_language)) and the [StarLogo TNG](https://en.wikipedia.org/wiki/StarLogo_TNG) [user interface](https://en.wikipedia.org/wiki/User_interface), that allowes users to [drag-and-drop](https://en.wikipedia.org/wiki/Drag-and-drop) visual objects to create an application that can run on the Android system, which runs on many mobile devices. [http://beta.appinventor.mit.edu](http://beta.appinventor.mit.edu/)

Android applications run in a [sandbox](https://en.wikipedia.org/wiki/Sandbox_(computer_security)), an isolated area of the operating system that does not have access to the rest of the system's resources, unless access permissions are granted by the user when the application is installed. Before installing an application, the [Play Store](https://en.wikipedia.org/wiki/Google_Play) displays all required permissions.

**Logcat**

**Logcat** is the command to view the internal log of the Android system.  Logs from various applications and portions of the system are collected, which then can be viewed and filtered by the **logcat** command. You can use **logcat** from an ADB shell to view the log messages. A good logcat is fried gold to a dev troubleshooting a problem.

**Using LogCat**

You can use LogCat from within DDMS or call it on an ADB shell.

[adb] logcat [<option>] ... [<filter-spec>] ...

The following table describes the **logcat** command line options:

|  |  |
| --- | --- |
| Option | Description |
| **-b <buffer>** | Loads an alternate log buffer for viewing, such as **event** or **radio**. The **main** buffer is used by default. |
| **-c** | Clears (flushes) the entire log and exits. |
| **-d** | Dumps the log to the screen and exits. |
| **-f <filename>** | Writes log message output to **<filename>**. The default is **stdout**. |
| **-g** | Prints the size of the specified log buffer and exits. |
| **-n <count>** | Sets the maximum number of rotated logs to **<count>**. The default value is 4. Requires the **-r**option. |
| **-r <kbytes>** | Rotates the log file every **<kbytes>** of output. The default value is 16. Requires the **-f** option. |
| **-s** | Sets the default filter spec to silent. |
| **-v <format>** | Sets the output format for log messages. The default is **brief** format. |

To export the terminal logcat to a text file use:

[adb] logcat > %location%/logcat.txt

**Filtering Log Output**

A note on **tags** and  **priorities:**

Every Android log message has a *tag* and a *priority* associated with it.

The tag of a log message is a short string indicating the system component from which the message originates (for example, "View" for the view system).

Priorities are different. These will tell you how serious the issue at hand is. The priorities are called by their letter code and are:  
  
**V Verbose – Lowest Priority  
D Debug  
I Info  
W Warning  
E Error  
F Fatal  
S Silent (suppress all output) – Highest Priority**  
These are in ascending order. In other words, Verbose or V is going to be the micro information which doesn't really mean much to anyone 99.99% of the time, where as Fatal or F is going to be a huge catastrophic issue. When filtering for a Priority it will include the Priority you give PLUS all HIGHER Priorities. So, for example, if you call to filter for Warning or W then it will give you Warning, Error, and Fatal.

Here's an example of logcat output that shows that the message relates to priority level "I" and tag "ActivityManager":

I/ActivityManager( 585): Starting activity: Intent { action=android.intent.action...}

To reduce the log output to a manageable level, you can restrict log output using *filter expressions*. Filter expressions let you indicate to the system the tags-priority combinations that you are interested in — the system suppresses other messages for the specified tags.

A filter expression follows this format **tag:priority**

Messages for that tag at or above the specified priority are written to the log. You can supply any number of **tag:priority** specifications in a single filter expression.

Here's an example of a filter expression that suppresses all log messages except those with the tag "ActivityManager", at priority "Info" or above, and all log messages with tag "MyApp", with priority "Debug" or above:

adb logcat ActivityManager:I MyApp:D \*:S

The final element in the above expression, **\*:S**, sets the priority level for all tags to "silent", thus ensuring only log messages with "View" and "MyApp" are displayed. Using **\*:S** is an excellent way to ensure that log output is restricted to the filters that you have explicitly specified — it lets your filters serve as a "whitelist" for log output.

The following filter expression displays all log messages with priority level "warning" and higher, on all tags:

adb logcat \*:W

**Controlling Log Output Format**

Log messages contain a number of metadata fields, in addition to the tag and priority. You can modify the output format for messages so that they display a specific metadata field. To do so, you use the **-v** option and specify one of the supported output formats listed below.

* **brief** — Display priority/tag and PID of the process issuing the message (the default format).
* **process** — Display PID only.
* **tag** — Display the priority/tag only.
* **raw** — Display the raw log message, with no other metadata fields.
* **time** — Display the date, invocation time, priority/tag, and PID of the process issuing the message.
* **threadtime** — Display the date, invocation time, priority, tag, and the PID and TID of the thread issuing the message.
* **long** — Display all metadata fields and separate messages with blank lines.

-v long after the logcat command formats the log so that it adds a date and time stamp to each line. It also separates each line with a blank line,making the log as a whole much easier to look through.

When starting LogCat, you can specify the output format you want by using the **-v** option:

[adb] logcat [-v <format>]

### Viewing Alternative Log Buffers

The Android logging system keeps multiple circular buffers for log messages, and not all of the log messages are sent to the default circular buffer. To see additional log messages, you can run the **logcat** command with the **-b** option, to request viewing of an alternate circular buffer. You can view any of these alternate buffers:

* **radio** — View the buffer that contains radio/telephony related messages.
* **events** — View the buffer containing events-related messages.
* **main** — View the main log buffer (default)

The usage of the **-b** option is:

[adb] logcat [-b <buffer>]

Here's an example of how to view a log buffer containing radio and telephony messages:

adb logcat -b radio

Using logcat for debugging from the bootlog:

* Boot into recovery.
* Open Terminal on your computer
* Enter the required logcat command. Preferably, -v long.
* In recovery tell it to reboot the phone. The logcat will start recording internally on your device at boot automatically.
* Once the phone is at the lockscreen let it sit for 5 minutes.
* Unlock the phone and let it sit for about 10 seconds.
* Restart the phone.
* Once you restart the phone open the logcat file on your desktop to make sure it’s not blank/empty/something went wrong and if everything’s golden - send to your favorite developer.

## What is a ROM?

A ROM image is a data file that contains information used on a Read Only Memory chip. For our purposes, that means a complete system image of a device. Each Android device has its own ROM image that contains files and code needed to boot the device up and run Android on it. But this is only part of a ROM. A ROM also contains a GUI (graphical user interface), required and useful applications, support files for those applications and the kernel. Let’s have a look at the parts that make a ROM.

# Parts of a ROM

## The kernel

Android (like many other Smartphone operating systems) runs on the Linux kernel.

Think of the kernel as an interface layer between the hardware and software on your device. The kernel decides when things happen, such as the LED indicator gets lit. An application sends a request to the operating system to blink the LED. The operating system then sends the request to the kernel, which makes the light flash for the amount of time requested by the OS.

This also keeps the application running in its own user-space and separate from the kernel. The kernel gives you the option to Force Close the application and the kernel can run untouched.

In a standard Android ROM, the kernel is bundled along with a set of instructions that tell the device how to load the kernel and the OS during boot. This is the boot.img that you see inside a zipped ROM that you’re not able to easily open. The device knows to extract this image to internal memory (the ramdisk) and follow a series of scripts (init scripts) to load the kernel and then the other portions of the OS. That’s what’s happening while you’re watching the boot animation. Interestingly enough this is done the same way for a PC, your smartphone, an Android tablet, or even a smart Linux powered toaster. If you’re feeling exceptionally geeky, plug your Android phone into the USB port on your PC and let the PC boot from the USB device. No, it doesn’t actually load, but you can watch the animation while it tries to match up the hardware support with what’s inside your PC. As I said, Linux is amazingly scalable and as a result so is Android.

## The operating system

Once the kernel is loaded, the init scripts tell the Operating System to load. Android is the user interface for a custom built Java virtual machine called Dalvik. Dalvik was written by Dan Bornstein, who named it after the fishing village of Dalvik in Iceland, where his family originated from. The debate of which Java VM is superior is best left for another discussion, so I’ll simply say that DalvikVM is a register-based machine versus true JavaVMs which are stack based.

The Dalvik machine creates executable files (.dex files) which can be interpreted by the OS and run by the end user. These .dex files are OS version dependant. That simply means that applications and core functions built to work with one version of Android may or may not work well with other versions. Google provides the tools through its Software Development Kit (SDK) for applications to communicate with the OS.

## Core functions

**Optional Functions**

**Fun Fact : XDA**

The first PDAs (Personal Digital Assistants) originally did not have any radio hardware. Soon they had bluetooth and infra-red connectivity but they did not evolve into "smartphones" until they got given radio for telephone and 2G data modem. So, O2, the British network provider (now part of Spain's Telefonica) contracted HTC to build such a device and so released the XDA (i.e. a PDA with the "X Factor" - telephony and 2G data hardware); hence the XDA was born in 2003. The "X" represents convergence of voice and information/data within one product; the "da" stands for "digital assistant", as in PDA.

**The Android File System:**

**Flash memory** is a [non-volatile](https://en.wikipedia.org/wiki/Non-volatile_memory) [computer storage](https://en.wikipedia.org/wiki/Computer_storage) chip that can be electrically erased and reprogrammed. Flash memory stores information in an array of memory cells made from [floating-gate transistors](https://en.wikipedia.org/wiki/Floating-gate_transistor). For storage applications NOR flash is not that great because it is not very dense (ie. not much storage per chip), is costly and is slow to write. NAND flash, on the other hand, is low cost, dense and writes fast, but has other limitations.  For NOR memory, reading and programming are random-access, and unlocking and erasing are block-wise. For NAND memory, reading and programming are page-wise, and unlocking and erasing are block-wise. In practice, flash file systems are only used for [memory technology devices](https://en.wikipedia.org/wiki/Memory_technology_device) (MTDs), which are embedded flash memories that do not have a controller because the controller and other features are provided by the flash file system. Removable flash [memory cards](https://en.wikipedia.org/wiki/Memory_card) and [USB flash drives](https://en.wikipedia.org/wiki/USB_flash_drive) have built-in controllers to perform wear leveling and error correction so use of a specific flash file system does not add any benefit. The file system on Android (YAFFS) is laid out over MTD partitions in your device's NAND flash (Internal memory), and the SD card. So, Android basically uses MTD (no controller) type NAND flash memory. The YAFFS provides the controller and other functions.

YAFFS stands for "yet another flash file system"(\*). As far as I am aware, YAFFS is the only file system, under any operating system, that has been designed specifically for use with NAND flash. YAFFS is thus designed to work within the constraints of, and exploit the features of, NAND flash to maximise performance. YAFFS uses journaling, error correction and verification techniques to overcome the areas where NAND typically fails, to enhance robustness. The result is a file system that exploits low-cost NAND chips and is both fast and robust. YAFFS is highly portable and runs under Linux, ucLinux and Windows CE. YAFFS is an open source project.

**As of Android 2.3 the Android system uses Ext4, the standard file system for Linux. Before that it used the YAFFS file system.** Some vendors replace the standard file system by their own one.

**Nandroid Backup**

A set of tools that will enable anyone who has root on their Android device to make FULL system backups, in case something goes wrong or you want to try out that new experimental ROM/theme. NANDroid will backup (and restore) **/system, /data, /cache, and /boot partitions.**

**AOSP vs AOKP**

AOSP is essentially what Google sends out (Android Open Source Project) as in completely stock. **VANILLA!**  
AOKP is a custom ROM that's based on AOSP with some mods baked into it. Also, the word "kang" is essentially stealing/copying.

**CLOCKWORKMOD BASED RECOVERY MENU**  
  
**1) Reboot Menu :**  
  
**reboot system now**  
  
This one is self-explanatory.  
  
**2) Install Menu :**  
  
**choose zip from (internal/ external) sdcard /**  
  
Lets you install any zip file (with any name) from any location on your SD card. The file can be for a ROM, a kernel, an application, a theme or any mod as long as it is in recovery-flashable zip format.   
  
**apply /sdcard/update.zip**  
  
This one is essentially the same as the ‘apply update from sdcard’ option of the main menu. widely used option for installing a ROM that you have downloaded and copied to your SD card. Entering this option will bring up a screen that will allow you to browse your SD card for the zip file.  
  
**apply update from sdcard**  
  
This can be used for installation of any official or unofficial update, ROM, kernel, theme etc. that is in a zip format installable from recovery, as long as the file is named update.zip and it has been placed on the root of your SD card (i.e. not in any sub-folder). Selecting this option will bring up a rather annoying confirmation prompt but this has saved us on multiple occasions from a lot of trouble we would have been into due to accidental key presses.  
  
**toggle signature verification**  
  
Turns the signature verification on and off. When signature verification is on, you will not be able to install any custom ROMs that haven’t been signed by the developers (most custom ROMs aren’t signed). Switching it off skips the signature verification check and proceeds with the installation.  
  
**toggle script asserts**  
  
Seldom-used option for a vast majority of users. It simply turns script asserts on or off. If you don’t know about these (I don’t), it’s best not to change this option.  
  
**3) Wipe Menu**  
  
**wipe data/factory reset**  
  
This option wipes all user data on the device as well as cache. Doing this will leave your phone in the state it was in when you bought it or when any custom ROM was first installed. It will also wipe any sd-ext partition that you might have setup. (see more about sd-ext below under partition)  
  
**wipe cache partition**  
  
This is a good practice to do this before flashing any ROM. The /cache partition just stores temporary files that are not critical to device operation and can be re-generated easily, this Wipes the cache partition of the device to clear all the data accumulated there over use. This is often used before installing a new ROM, app, kernel or any similar mod via recovery.  
  
**Wipe Dalvik Cache**  
  
Allows you to wipe the cache for the Dalvik virtual machine. The dalvik cache wipe is quite similar to cache wipe but it stores the post ran java applications. Since Android is JAVA based, it uses the same java virtual machine for compiling. The dalvik cache just stores post-compiled applications in order to speed up the system. Wiping this just forces the system to re-cache those application. It causes no problems but a slight hint of lag on first boot. This is required before most ROM installations and at other occasions too, for fixing some problems.  
  
**Wipe Battery Stats**  
  
Wipes the saved battery usage statistics and effectively recalibrates the battery. Useful in various scenarios when Android isn’t showing correct battery levels.  
  
**4) Nandroid menu**  
  
backup and restore Undoubtedly one of the most important features provided by a custom recovery, the backup and restore feature – also known as Nandroid backup – allows you to take a snapshot of your phone’s entire internal memory including all partitions, and save it on the SD card.   
  
**Backup**  
  
Takes a Nandroid backup, as explained above.  
  
**Restore**  
  
Lets you restore a previously taken backup. Entering this option presents you with a list of existing backups from the SD card that you can choose from for restoration.  
  
**Advanced Restore (new options are available separately to restore from external or internal SDcard in the latest CWM)**  
  
This option is similar to the Restore option but once a backup has been selected to be restored, this option allows you to choose what parts of it to restore. You can choose to restore the boot, system, data, cache and sd-ext partitions.  
  
**5) Storage menu**  
  
**mounts and storage**  
  
Allows you to perform maintenance tasks on all the internal and external partitions of your android device  
  
**mount/unmount /system, /data, /cache, /sdcard, /emmc.**  
  
These options let you toggle between mounting or unmounting these respective partitions. Most users don’t need to change these options.  
  
**format system, data, cache, sdcard or sd-ext**  
  
These let you directly format any of these partitions. **Take extreme care**with this option as formatting any of these partitions will result in losing all data on them, especially the boot and system partitions. Formatting the system partition will remove your ROM and leave your phone without an operating system while wiping the boot partition may brick your phone unless you restore or flash another one before rebooting your device. See below more explanation about these partitions.   
  
**mount USB storage**  
  
Lets you enable USB mass storage mode for your SD card right from recovery so that you can connect it to your computer via USB and transfer any files to/from it without having to leave recovery.  
  
**6) Advanced**  
  
This section contains a few options most users will not require, Here are the options from this section:   
  
**Report Error**  
  
In case of errors, this feature can be used to save a log of recent ClockworkMod recovery operations on the SD card that you can later report from Android using ROM Manager.  
  
**Key Test**  
  
Lets you press any of the hardware keys to see if they are properly functioning, and to see their key codes.  
  
**Partition SD Card**  
  
This option gives you a no-frills way to partition your SD card properly for use with ROMs that support data2ext (a very handy hack for low internal memory devices that enables an /sd-ext partition on the SD card to be used as the internal user data storage i.e. as the /data partition). Once this option is selected, you will be given options to choose the sizes for the /sd-ext partition as well as an optional /swap partition on the SD card, and will then automatically format it for you, leaving the remaining space for normal SD card usage. **This option will wipe all data from your SD card so use it with caution!**  
  
**Fix Permissions**  
  
Fixes the file permissions for the internal memory partitions back to default. This is very useful as a fix for several errors and Force-Closes that start appearing after you or an application you installed and provided root access end up messing up the permissions of important files.  
  
**PARTITIONS :**  
  
The Android uses several partitions to organize files and folders on the device. Each of these partitions has a distinct role in the functionality of the device, but not many Android users know the significance of each partition and its contents. In this guide, we will take you on a tour of Android partitions, what they contain and what can be the possible consequences of modifying their content.  
Let’s start with a list of standard internal memory partitions on Android phones and tablets. These are:  
  
• /boot  
• /system  
• /recovery  
• /data  
• /cache  
• /misc  
  
In addition, there are the SD card partitions.  
  
• /sdcard  
• /sd-ext  
  
Note that only /sdcard is found in all Android devices and the rest are present only in select devices. Let’s now take a look at the purpose and contents of each of these partitions.  
  
**/boot**  
  
This is the partition that enables the phone to boot, as the name suggests. It includes the bootloader and the kernel. Without this partition, the device will simply not be able to boot. Wiping this partition from recovery should only be done if absolutely required and once done, the device must NOT be rebooted before installing a new one, which can be done by installing a ROM that includes a /boot partition.  
  
**/system**  
  
This partition basically contains the entire operating system, other than the kernel and the bootloader. This includes the Android user interface as well as all the system applications that come pre-installed on the device. Wiping this partition will remove Android from the device without rendering it unbootable, and you will still be able to put the phone into recovery or bootloader mode to install a new ROM.  
  
**/recovery**  
  
The recovery partition can be considered as an alternative boot partition that lets you boot the device into a recovery console for performing advanced recovery and maintenance operations on it. We have already learnt about this partition and its contents above.  
  
**/data**  
  
Also called userdata, the data partition contains the user’s data – this is where your contacts, messages, settings and apps that you have installed go. Wiping this partition essentially performs a factory reset on your device, restoring it to the way it was when you first booted it, or the way it was after the last official or custom ROM installation. When you perform a wipe data/factory reset from recovery, it is this partition that you are wiping.  
  
**/cache**  
  
This is the partition where Android stores frequently accessed data and app components. Wiping the cache doesn’t effect your personal data but simply gets rid of the existing data there, which gets automatically rebuilt as you continue using the device.  
  
**/misc**  
  
This partition contains miscellaneous system settings in form of on/off switches. These settings may include CID (Carrier or Region ID), USB configuration and certain hardware settings etc. This is an important partition and if it is corrupt or missing, several of the device’s features will will not function normally.  
  
**/sdcard**  
  
This is not a partition on the internal memory of the device but rather the SD card. In terms of usage, this is your storage space to use as you see fit, to store your media, documents, ROMs etc. on it. Wiping it is perfectly safe as long as you backup all the data you require from it, to your computer first. Though several user-installed apps save their data and settings on the SD card and wiping this partition will make you lose all that data.  
  
On devices with both an internal and an external SD card – devices like the Samsung Galaxy SII – the /sdcard partition is always used to refer to the internal SD card. For the external SD card – if present – an alternative partition is used, which differs from device to device. In case of Samsung Galaxy S series devices, it is /sdcard/External\_sd while in many other devices, it is /sdcard2. Unlike /sdcard, no system or app data whatsoever is stored automatically on this external SD card and everything present on it has been added there by the user. You can safely wipe it after backing up any data from it that you need to save.  
  
**/sd-ext**  
  
This is not a standard Android partition, but has become popular in the custom ROM scene. It is basically an additional partition on your SD card that acts as the /data partition when used with certain ROMs that have special features called APP2SD+ or data2ext enabled. It is especially useful on devices with little internal memory allotted to the /data partition. Thus, users who want to install more programs than the internal memory allows can make this partition and use it with a custom ROM that supports this feature, to get additional storage for installing their apps. Wiping this partition is essentially the same as wiping the /data partition – you lose your contacts, SMS, market apps and settings.

**ANDROID CPU GOVERNORS:**

The Governor is a piece of code/program usually implemented in the kernel that interacts with the device hardware to increase or decrease the processor's(CPU’s) clock speed(frequency) (e.g. at low usage, it will set the processor speed to 400 MHz, but as usage increases, it would scale up to 1000 MHz).

**NOTE:**You cannot change your CPU governor unless your phone is rooted and you have a ROM or app (Eg. SetCPU) that lets you make a change. Also, different kernels (the intermediary software between your phone's hardware and the operating system) offer different sets of governors.