ANDROID FORENSICS

Reading Material:

Android Forensics

Andrew Hoog

In this lecture

Why Android?

Introduction to Android

Booting

Data Storage

Android Partitions and Filesystems

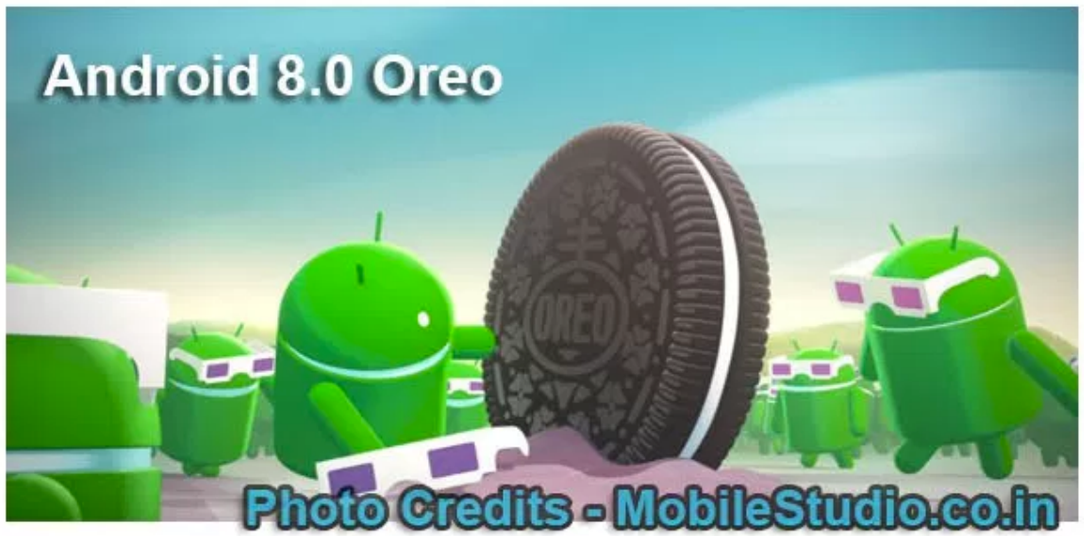
Reading: Android Forensics by Andrew Hoog

- Why Android?
- Introduction to Android
- Booting
- Data Storage
- Android Partitions and Filesystems

WHY ANDROID FORENSICS?

- jewellery - watches, rings

- glasses
- clothes
- cars
- homes & appliances
- cameras (eg polaroid)



https://i0.wp.com/mobilestudio.co.in/wp-content/uploads/2017/09/Android-Versions-History.jpg?fit=1456%2C896

VERSIONS

Code name	Version	API level
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(no code name)	1.1	API level 2
(no code name)	1.0	API level 1

In Android 8.0 (Oreo) and higher, individual builds are identified with the build ID format **PVBB.YYMMDD.bbb[.Cn]**, where:

- P represents the first letter of the code name of the platform release, e.g. O is Oreo.
- V represents a supported vertical. By convention, P represents the primary platform branch.
- BB represents an alphanumeric code that allows Google to identify the exact code branch the build was made from.
- YYMMDD identifies the date when the release is branched from or synced with the development branch. It is not guaranteed to be the exact date at which a build was made as it is common for minor variations added to an existing build to re-use the same date code as the existing build.
- bbb identifies individual versions related to the same date code, sequentially starting with 001.
- Cn is an optional, alphanumeric that identifies a hotfix on top of an existing PVBB.YYMMDD.bbb build, starting from A1.

THE ANDROID STACK

ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL

CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM

AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT

CORE LIBRARIES • ART • DALVIK VM

AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV

DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

ANDROID ARCHITECTURE

framework is used most often by application developers

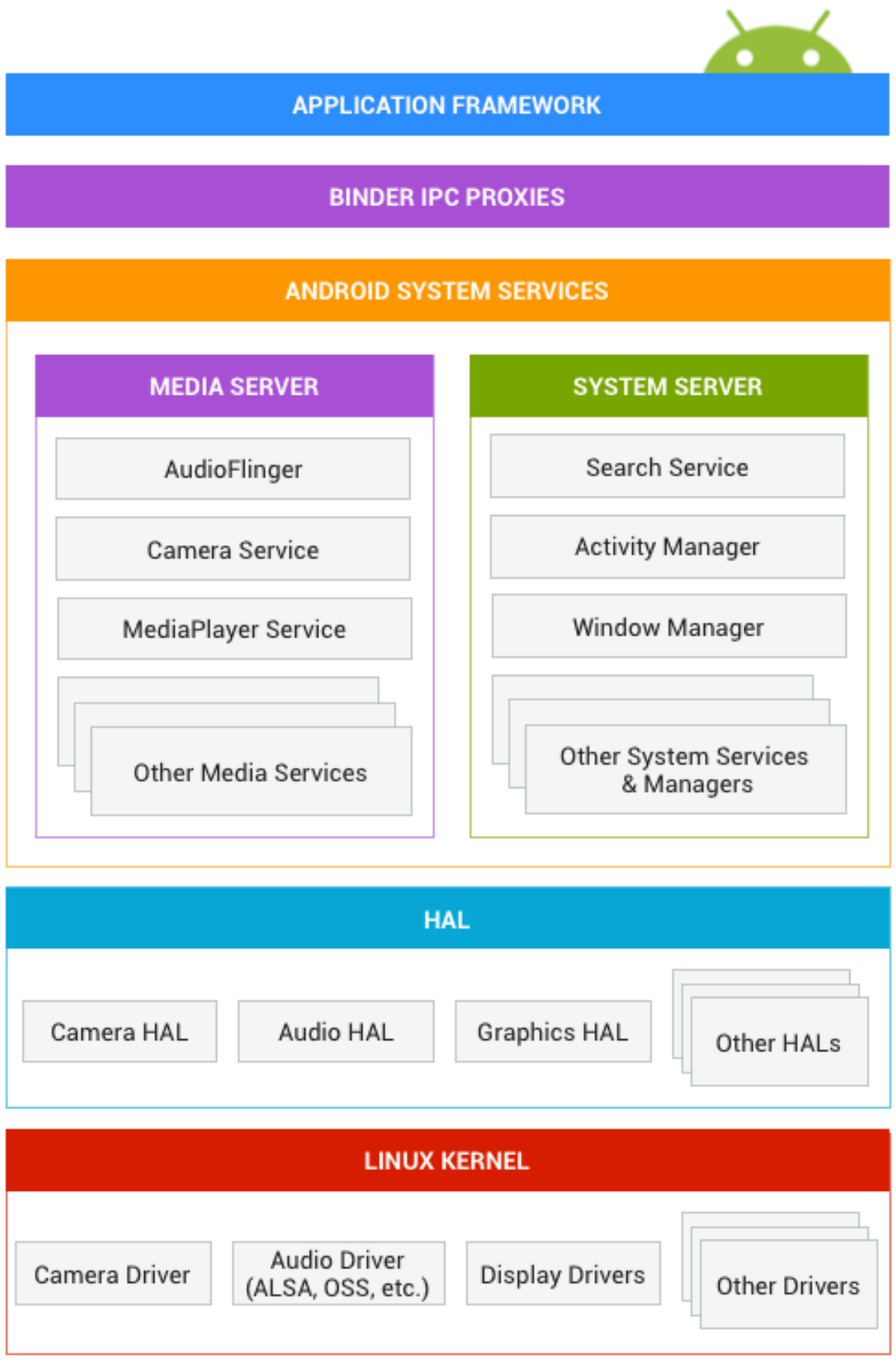
enables high level framework APIs to interact with Android system services

functionality exposed by application framework APIs communicates with system services to access the underlying hardware

Hardware Abstraction Layer (HAL) defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations

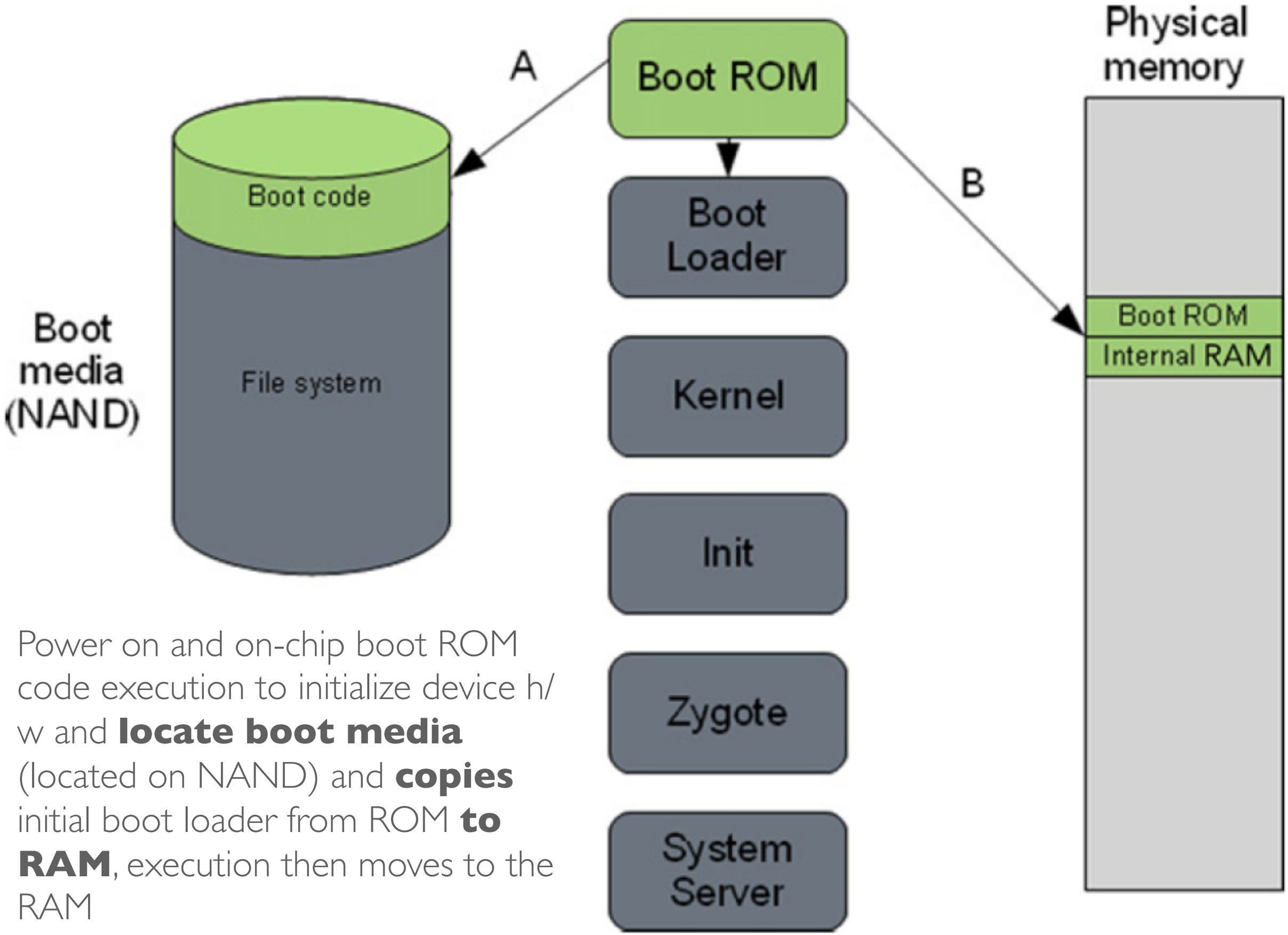
Architecture

Android system architecture contains the following components:

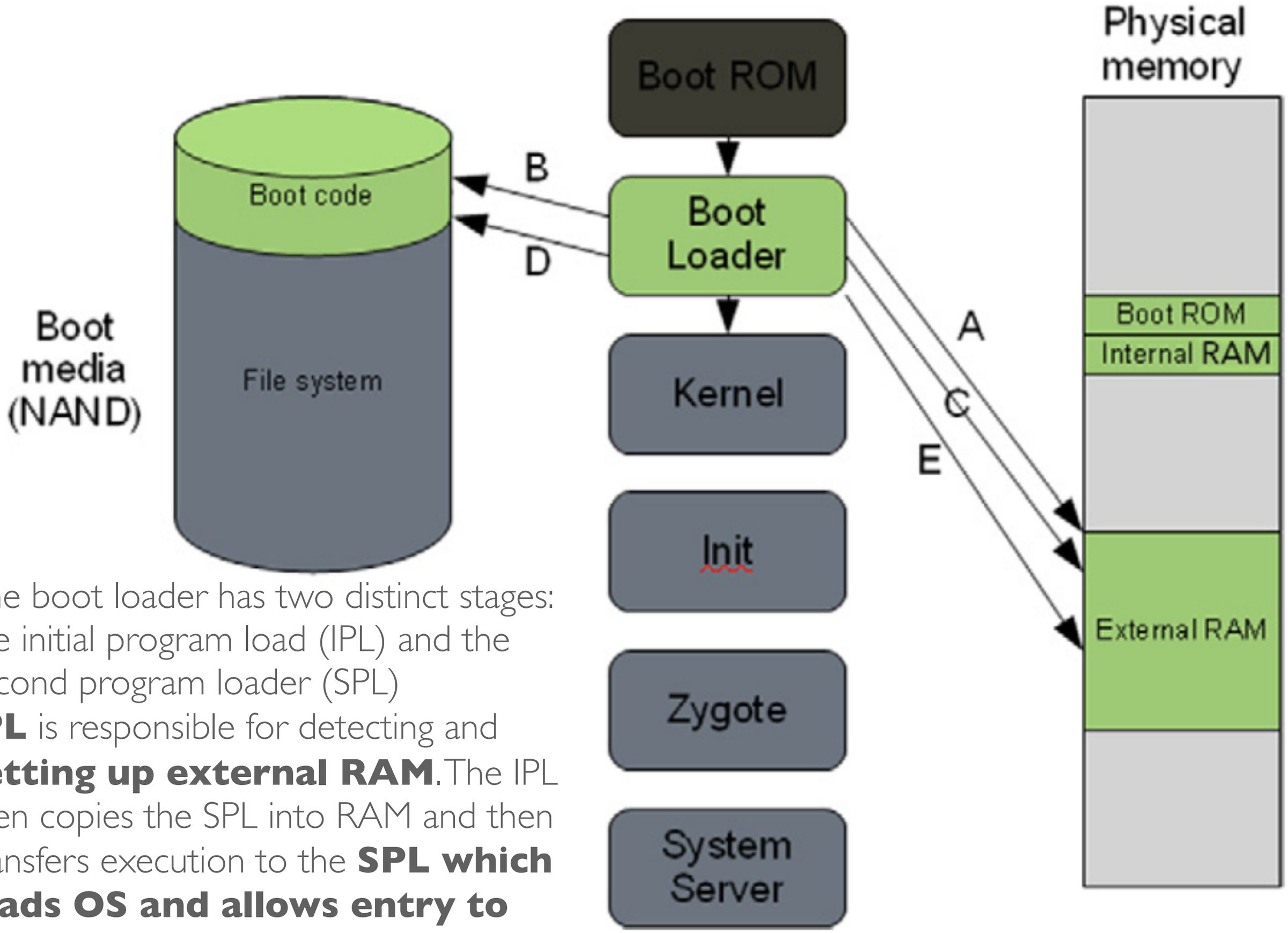


ANDROID BOOT PROCESS

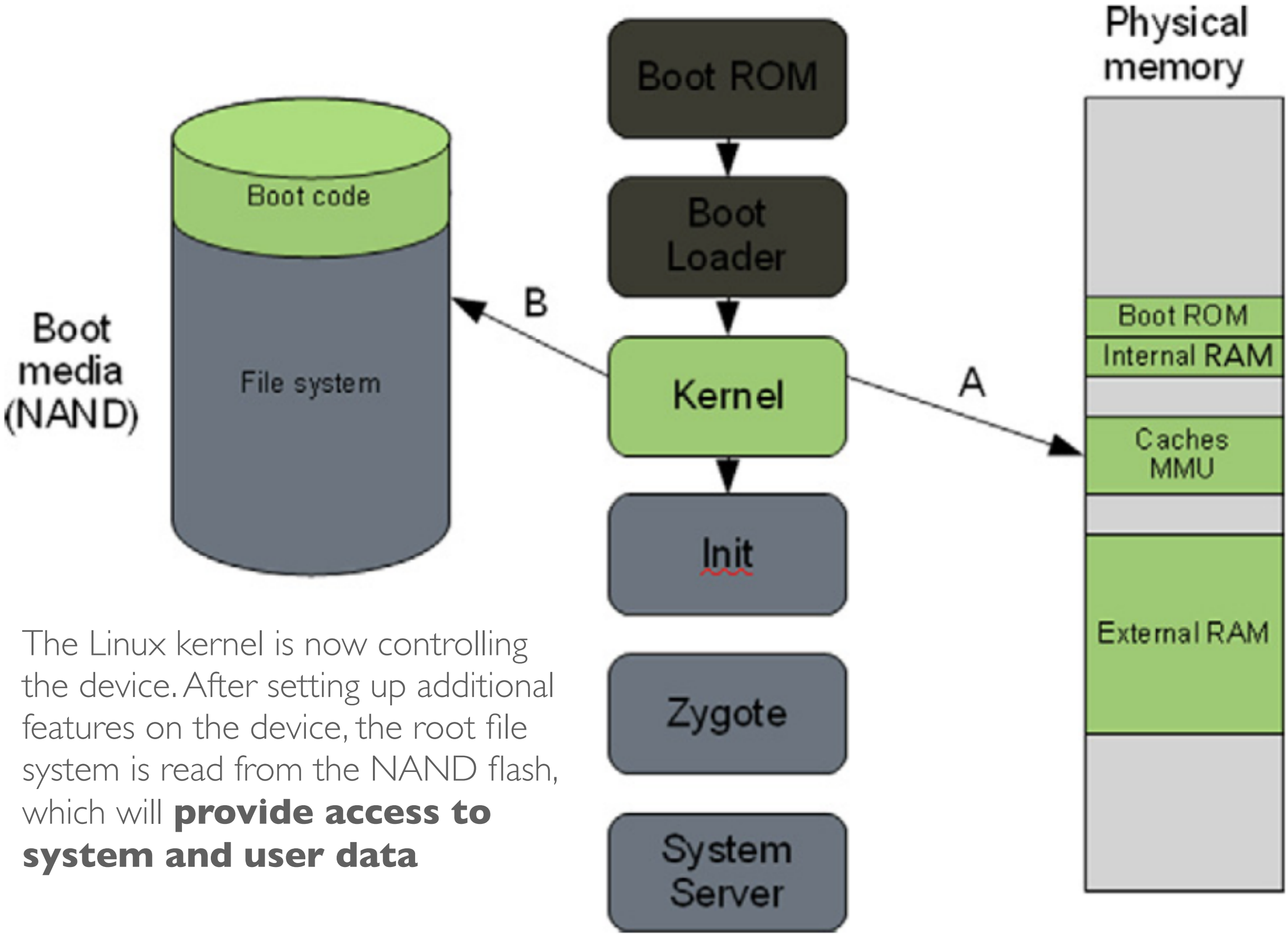
1. Power on and on-chip boot ROM code execution
2. The boot loader
3. The Linux kernel
4. The init process
5. Zygote and Dalvik
6. The system server
7. Boot complete



Power on and on-chip boot ROM code execution to initialize device h/w and **locate boot media** (located on NAND) and **copies** initial boot loader from ROM **to RAM**, execution then moves to the RAM

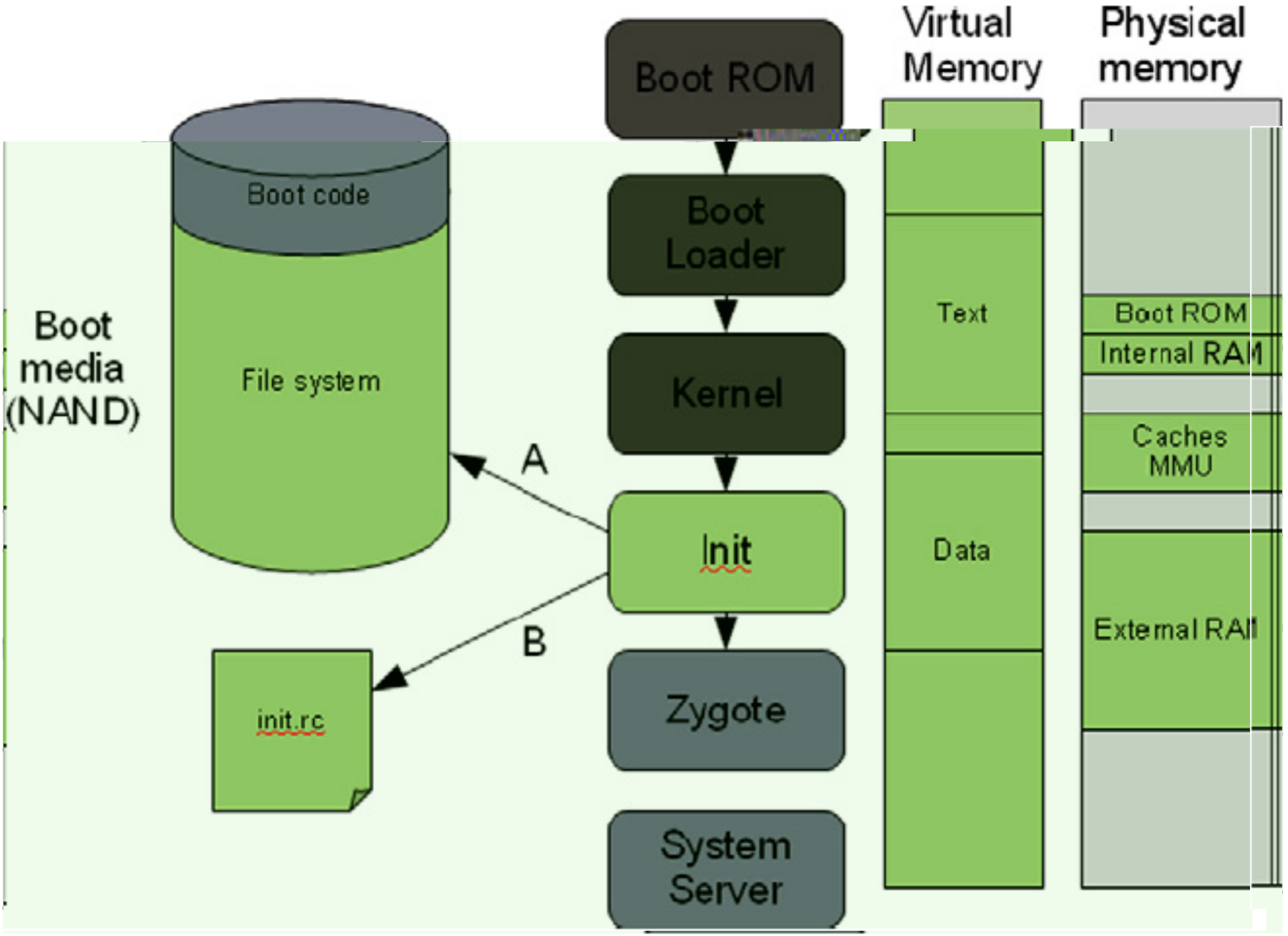


The boot loader has two distinct stages: the initial program load (IPL) and the second program loader (SPL)
IPL is responsible for detecting and **setting up external RAM**.The IPL then copies the SPL into RAM and then transfers execution to the **SPL which loads OS and allows entry to other boot modes** such as fastboot

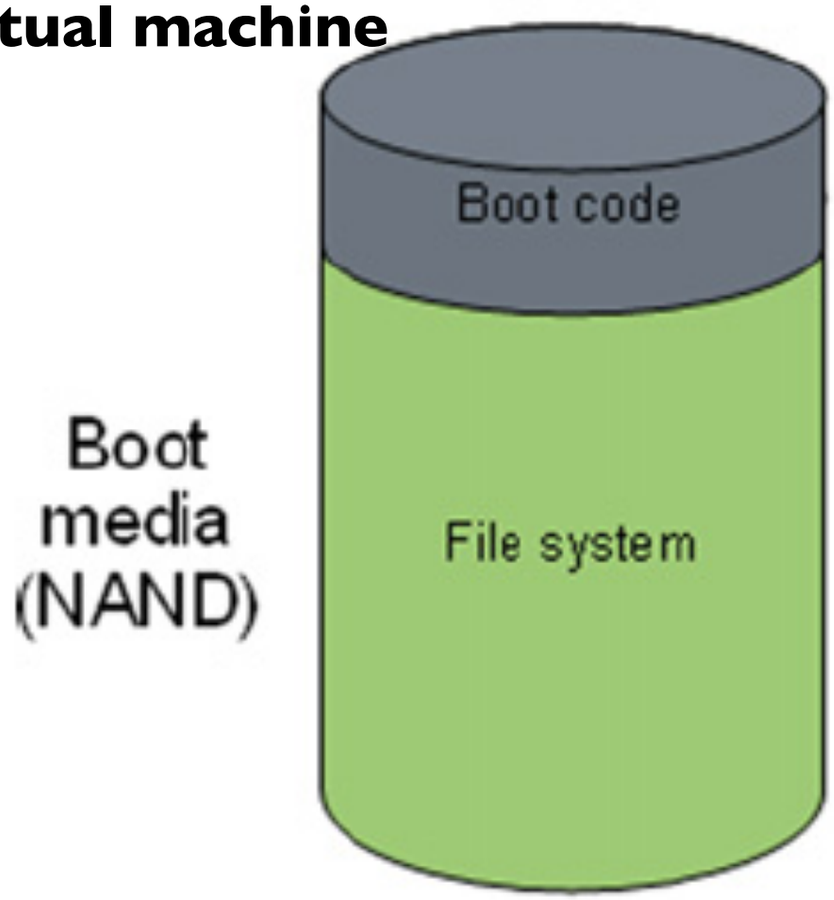


The Linux kernel is now controlling the device. After setting up additional features on the device, the root file system is read from the NAND flash, which will **provide access to system and user data**

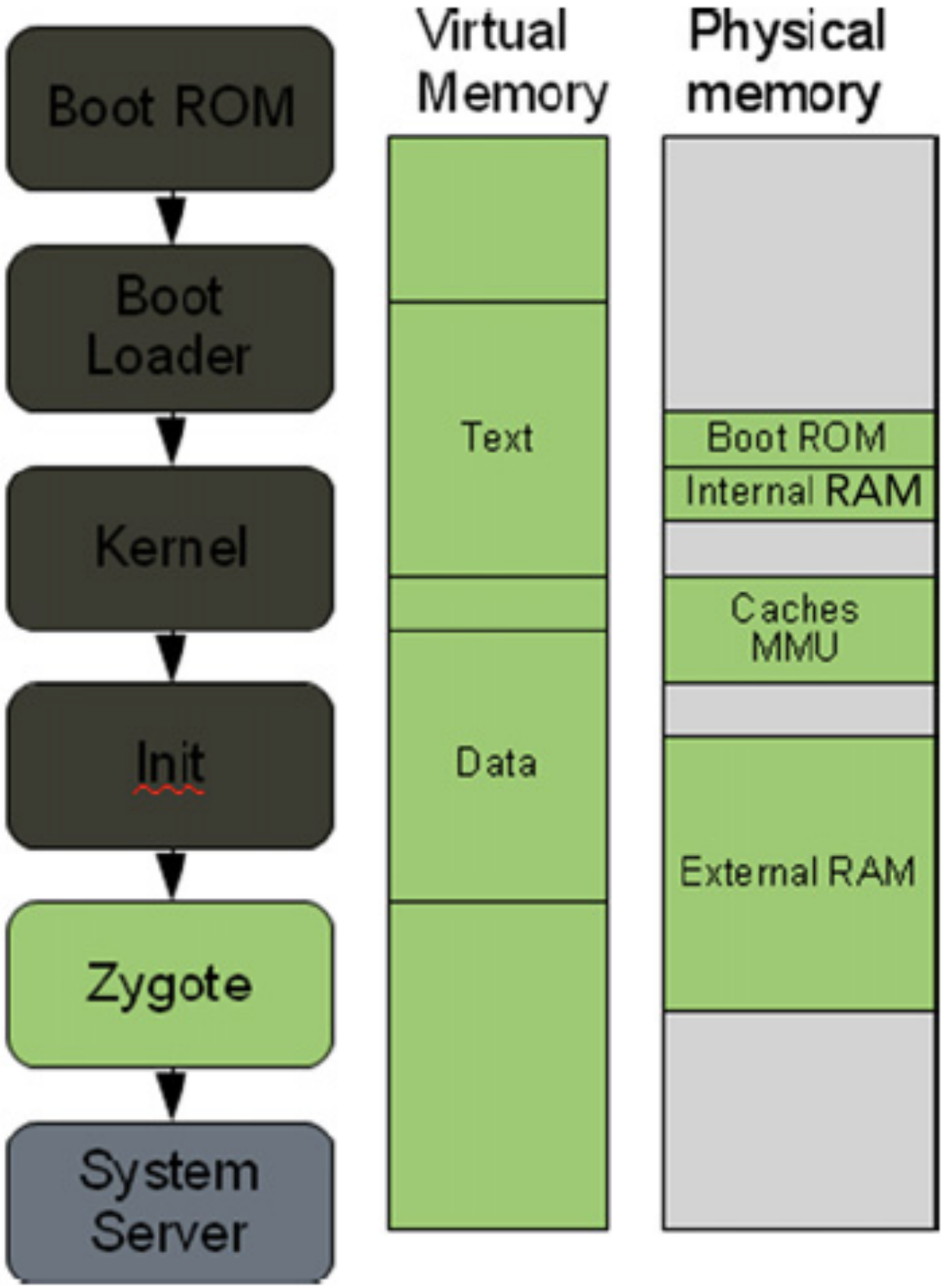
Once the kernel has access to the system partition, it can process the **init** scripts that start key system and user processes



The **Zygote** sequence essentially sets up the Java runtime environment and registers a socket with the system; hence, new applications that need to initialize can request a new **Dalvik virtual machine**



The system finally sends a standard broadcast action called **ACTION_BOOT_COMPLETED**, which alerts dependent processes that the boot process is complete. The Android system is now fully operational and is ready to interact with the user. Core activity can now occur.



STORAGE

1. NAND
    - Data storage
    - Logs
    - File systems
2. Ram

STORAGE - APPLICATIONS

- When an application is installed, an internal data storage is created in a subdirectory of
  - /data/data
- eg: the Android browser is com.android.browser
- The data files are stored in
- /data/data/com.android.browser

STORAGE - APPLICATIONS

- Common subdirectories found here include
  - lib - custom library files an application requires
  - files - files the developer saves to internal storage
  - cache - files cached by the application
  - databases - SQLite database and journal files

STORAGE - DEVELOPERS

- Shared preferences - primitive data (boolean, int, float, long, string) types for key value pairs
    - found at /data/data/com.android.browser/shared_prefs
- Internal storage - /data/data
- External storage - usually FAT32 mounted at /mnt/
- SQLite - a single cross platform file
    - found at /data/data/com.android.browser/databases
- Network & Cloud

LOGS - KERNEL

- dmesg command prints all available kernel messages, good for timestamps, activities, booting up etc (USB debugging must be enabled for it to work)

- logcat - system and application logs (V verbose, D debug, I information, E error, W warning, F fatal, S silent)

- logcat -b radio shows cellular radio debug can give times, IPs, wireless carrier info, SMS info

LOGS - OS

- dumpsys
  - currently running services, dump of each service, process information, services, broadcasts
  - service account - gives all accounts on phone, eg FB, gmail etc
  - iphonesubinfo - not iphone humourous name instead
    - returns phone type and mobile equipment device identifier
  - location - last known location
- dumpstate -
  - stack traces, device info, system info, memory, cpu, system log, event log, radio log
- **bugreport - combines logcat dumpsys and dumpstate**

```
[ 5086.751975] ieee80211 phy0: hwsim hw_scan request
[ 5086.751980] ieee80211 phy0: hw scan 5180 MHz
[ 5086.872663] ieee80211 phy0: hw scan 5200 MHz
[ 5086.992041] ieee80211 phy0: hw scan 5220 MHz
[ 5087.111778] ieee80211 phy0: hw scan 5240 MHz
[ 5087.232248] ieee80211 phy0: hw scan 5260 MHz
[ 5087.353629] ieee80211 phy0: hw scan 5280 MHz
[ 5087.470686] ieee80211 phy0: hw scan 5300 MHz
[ 5087.591274] ieee80211 phy0: hw scan 5320 MHz
[ 5087.715179] ieee80211 phy0: hw scan 5500 MHz
[ 5087.831123] ieee80211 phy0: hw scan 5520 MHz
[ 5087.953309] ieee80211 phy0: hw scan 5540 MHz
[ 5088.074076] ieee80211 phy0: hw scan 5560 MHz
[ 5088.191284] ieee80211 phy0: hw scan 5580 MHz
[ 5088.311381] ieee80211 phy0: hw scan 5600 MHz
[ 5088.430871] ieee80211 phy0: hw scan 5620 MHz
[ 5088.554014] ieee80211 phy0: hw scan 5640 MHz
[ 5088.670389] ieee80211 phy0: hw scan 5660 MHz
[ 5088.793998] ieee80211 phy0: hw scan 5680 MHz
[ 5088.913592] ieee80211 phy0: hw scan 5700 MHz
[ 5089.031219] ieee80211 phy0: hw scan 5745 MHz
[ 5089.151409] ieee80211 phy0: hw scan 5765 MHz
[ 5089.273303] ieee80211 phy0: hw scan 5785 MHz
[ 5089.391097] ieee80211 phy0: hw scan 5805 MHz
[ 5089.511676] ieee80211 phy0: hw scan 5825 MHz
[ 5089.630376] ieee80211 phy0: hw scan complete
[ 5089.630933] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5089.631431] ieee80211 phy0: mac80211_hwsim_get_survey (idx=0)
[ 5089.631436] ieee80211 phy0: mac80211_hwsim_get_survey (idx=1)
[ 5089.733544] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5102.379889] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5102.503562] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5108.498773] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5108.591502] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5120.081439] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5120.181775] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5151.041452] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5151.142144] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5162.378196] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5162.504312] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5167.602081] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5167.703510] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5169.063929] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5169.162115] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5169.462153] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5169.723555] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5172.571320] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5172.671235] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5203.121304] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5203.223070] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5222.377319] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5222.503387] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5227.584951] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5227.680768] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5228.221077] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5228.320969] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5228.381188] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5228.481758] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5229.013160] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5229.112864] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
[ 5232.594608] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=0 smps=auto)
[ 5232.691005] ieee80211 phy0: mac80211_hwsim_config (freq=0 idle=0 ps=1 smps=auto)
```

Channel Frequencies supported by wifi registered to mac80211 (often called cfg80211 instead)

```
331  static const struct ieee80211_channel hwsim_channels_5ghz[] = {
332          CHAN5G(5180), /* Channel 36 */
333          CHAN5G(5200), /* Channel 40 */
334          CHAN5G(5220), /* Channel 44 */
335          CHAN5G(5240), /* Channel 48 */
336
337          CHAN5G(5260), /* Channel 52 */
338          CHAN5G(5280), /* Channel 56 */
339          CHAN5G(5300), /* Channel 60 */
340          CHAN5G(5320), /* Channel 64 */
341
342          CHAN5G(5500), /* Channel 100 */
343          CHAN5G(5520), /* Channel 104 */
344          CHAN5G(5540), /* Channel 108 */
345          CHAN5G(5560), /* Channel 112 */
346          CHAN5G(5580), /* Channel 116 */
347          CHAN5G(5600), /* Channel 120 */
348          CHAN5G(5620), /* Channel 124 */
349          CHAN5G(5640), /* Channel 128 */
350          CHAN5G(5660), /* Channel 132 */
351          CHAN5G(5680), /* Channel 136 */
352          CHAN5G(5700), /* Channel 140 */
353
354          CHAN5G(5745), /* Channel 149 */
355          CHAN5G(5765), /* Channel 153 */
356          CHAN5G(5785), /* Channel 157 */
357          CHAN5G(5805), /* Channel 161 */
358          CHAN5G(5825), /* Channel 165 */
359          CHAN5G(5845), /* Channel 169 */
360  };
```

torvalds / linux Watch 6,651 Star 69,040 Fork 24,838

Code Pull requests 249 Projects 0 Insights

Branch: master linux / drivers / net / wireless / mac80211_hwsim.c Find file Copy path

jmberg mac80211_hwsim: check that n_limits makes sense 9c5d3af 19 days ago

94 contributors and others

3953 lines (3336 sloc) | 105 KB Raw Blame History

```
1  /*
2   * mac80211_hwsim - software simulator of 802.11 radio(s) for mac80211
3   * Copyright (c) 2008, Jouni Malinen <j@w1.fi>
4   * Copyright (c) 2011, Javier Lopez <jlopex@gmail.com>
5   * Copyright (c) 2016 - 2017 Intel Deutschland GmbH
6   * Copyright (C) 2018 Intel Corporation
7   *
```

```
1619 static int mac80211_hwsim_config(struct ieee80211_hw *hw, u32 changed)
1620 {
1621         struct mac80211_hwsim_data *data = hw->priv;
1622         struct ieee80211_conf *conf = &hw->conf;
1623         static const char *smps_modes[IEEE80211_SMPS_NUM_MODES] = {
1624                 [IEEE80211_SMPS_AUTOMATIC] = "auto",
1625                 [IEEE80211_SMPS_OFF] = "off",
1626                 [IEEE80211_SMPS_STATIC] = "static",
1627                 [IEEE80211_SMPS_DYNAMIC] = "dynamic",
1628         };
1629         int idx;
1630
1631         if (conf->chandef.chan)
1632                 wiphy_dbg(hw->wiphy,
1633                           "%s (freq=%d(%d - %d)/%s idle=%d ps=%d smps=%s)\n",
1634                           __func__,
1635                           conf->chandef.chan->center_freq,
1636                           conf->chandef.center_freq1,
1637                           conf->chandef.center_freq2,
1638                           hwsim_chanwidths[conf->chandef.width],
1639                           !!(conf->flags & IEEE80211_CONF_IDLE),
1640                           !!(conf->flags & IEEE80211_CONF_PS),
1641                           smps_modes[conf->smps_mode]);
1642         else
1643                 wiphy_dbg(hw->wiphy,
1644                           "%s (freq=0 idle=%d ps=%d smps=%s)\n",
1645                           __func__,
1646                           !!(conf->flags & IEEE80211_CONF_IDLE),
1647                           !!(conf->flags & IEEE80211_CONF_PS),
1648                           smps_modes[conf->smps_mode]);
1649
1650         data->idle = !!(conf->flags & IEEE80211_CONF_IDLE);
1651
1652         WARN_ON(conf->chandef.chan && data->use_chanctx);
1653
1654         mutex_lock(&data->mutex);
1655         if (data->scanning && conf->chandef.chan) {
1656                 for (idx = 0; idx < ARRAY_SIZE(data->survey_data); idx++) {
1657                         if (data->survey_data[idx].channel == data->channel) {
1658                                 data->survey_data[idx].start =
```

```
1559                                         data->survey_data[idx].next_start;
1560                                 data->survey_data[idx].end = jiffies;
1561                                 break;
1562                         }
1563                 }
1564
1565                 data->channel = conf->chandef.chan;
1566
1567                 for (idx = 0; idx < ARRAY_SIZE(data->survey_data); idx++) {
1568                         if (data->survey_data[idx].channel &&
1569                             data->survey_data[idx].channel != data->channel)
1570                                 continue;
1571                         data->survey_data[idx].channel = data->channel;
1572                         data->survey_data[idx].next_start = jiffies;
1573                         break;
1574                 }
1575         } else {
1576                 data->channel = conf->chandef.chan;
1577         }
1578         mutex_unlock(&data->mutex);
1579
1580         if (!data->started || !data->beacon_int)
1581                 tasklet_hrtimer_cancel(&data->beacon_timer);
1582         else if (!hrtimer_is_queued(&data->beacon_timer.timer)) {
1583                 u64 tsf = mac80211_hwsim_get_tsf(hw, NULL);
1584                 u32 bcn_int = data->beacon_int;
1585                 u64 until_tbtt = bcn_int - do_div(tsf, bcn_int);
1586
1587                 tasklet_hrtimer_start(&data->beacon_timer,
1588                                       ns_to_ktime(until_tbtt * 1000),
1589                                       HRTIMER_MODE_REL);
1590         }
1591
1592         return 0;
1593 }
1594
```

https://github.com/torvalds/linux/blob/master/drivers/net/wireless/mac80211_hwsim.c

```
1103  static int mac80211_hwsim_config(struct ieee80211_hw *hw, u32 changed)
1104  {
1105          struct mac80211_hwsim_data *data = hw->priv;
1106          struct ieee80211_conf *conf = &hw->conf;
1107          static const char *smps_modes[IEEE80211_SMPS_NUM_MODES] = {
1108                  [IEEE80211_SMPS_AUTOMATIC] = "auto",
1109                  [IEEE80211_SMPS_OFF] = "off",
1110                  [IEEE80211_SMPS_STATIC] = "static",
1111                  [IEEE80211_SMPS_DYNAMIC] = "dynamic",
1112          };
1113
1114          if (conf->chandef.chan)
1115                  wiphy_debug(hw->wiphy,
1116                              "%s (freq=%d(%d - %d)/%s idle=%d ps=%d smps=%s)\n",
1117                              __func__,
1118                              conf->chandef.chan->center_freq,
1119                              conf->chandef.center_freq1,
1120                              conf->chandef.center_freq2,
1121                              hwsim_chanwidths[conf->chandef.width],
1122                              !!(conf->flags & IEEE80211_CONF_IDLE),
1123                              !!(conf->flags & IEEE80211_CONF_PS),
1124                              smps_modes[conf->smps_mode]);
1125          else
1126                  wiphy_debug(hw->wiphy,
1127                              "%s (freq=0 idle=%d ps=%d smps=%s)\n",
1128                              __func__,
1129                              !!(conf->flags & IEEE80211_CONF_IDLE),
1130                              !!(conf->flags & IEEE80211_CONF_PS),
1131                              smps_modes[conf->smps_mode]);
1132
1133          data->idle = !!(conf->flags & IEEE80211_CONF_IDLE);
1134
1135          data->channel = conf->chandef.chan;
1136
1137          WARN_ON(data->channel && channels > 1);
1138
1139          data->power_level = conf->power_level;
1140          if (!data->started || !data->beacon_int)
1141                  tasklet_hrtimer_cancel(&data->beacon_timer);
1142          else if (!hrtimer_is_queued(&data->beacon_timer.timer)) {
1143                  u64 tsf = mac80211_hwsim_get_tsf(hw, NULL);
1144                  u32 bcn_int = data->beacon_int;
1145                  u64 until_tbtt = bcn_int - do_div(tsf, bcn_int);
1146
1147                  tasklet_hrtimer_start(&data->beacon_timer,
1148                                        ns_to_ktime(until_tbtt * 1000),
1149                                        HRTIMER_MODE_REL);
1150          }
1151
1152          return 0;
1153  }
1154
```

https://android.googlesource.com/kernel/common/+/bcmdhd-3.10/drivers/net/wireless/mac80211_hwsim.c

CFG80211 OR MAC80211

- Contains an algorithm for processing regulatory rules and applying them
- Has regulatory compliance built in
- Provides an API for 80211 drivers (cfg80211 or mac80211 drivers)
- Linux wireless drivers should be written targeting either cfg80211 for fullmac devices or mac80211 for softmac devices (fullmac or hardmac implements the MAC layer in hardware, softmac doesn't)
- Upon initialisation of the wireless core (cfg80211) a world regulatory domain (highly restrictive) will be set as the central regulatory domain (CRDA)
- When a wiphy is registered to cfg80211 the effective set of permitted channels/maximum power is calculated for that device based on the current regulatory domain

CFG80211 OR MAC80211

- Softmac advantages
  - Potentially lower hardware costs
  - Possibility to upgrade to newer standards by updating the driver only
  - Possibility to correct faults in the MAC implementation by updating the driver only
- Hardmac advantage is that since the MAC functions are implemented in hardware, they contribute less CPU load
- Device registers with cfg80211, a list of supported bands, each band has a list of supported channels.
- During registration, cfg80211, ensures only the allowed channels for the currently set regulatory domain will be left enabled.

DMESG HEALTH: BATTERY

```
[ 5102.379889] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
[ 5102.503562] healthd: battery l=100 v=5000 t=25.0 h=2 st=2 c=900 fc=300000 cc=10 chg=a
```

- l =100% (level)
- v=500 (voltage)
- t=25 degrees (temp)
- fc = fullcharge
- cc =cycle count
- chg=a (charger = a for AC, u for USB or w for Wireless)

```
295     if (logthis) {
296         char dmesgline[256];
297         size_t len;
298         if (props.batteryPresent) {
299             snprintf(dmesgline, sizeof(dmesgline),
300                  "battery l=%d v=%d t=%s%d.%d h=%d st=%d",
301                  props.batteryLevel, props.batteryVoltage,
302                  props.batteryTemperature < 0 ? "-" : "",
303                  abs(props.batteryTemperature / 10),
304                  abs(props.batteryTemperature % 10), props.batteryHealth,
305                  props.batteryStatus);
306
307             len = strlen(dmesgline);
308             if (!mHealthdConfig->batteryCurrentNowPath.isEmpty()) {
309                 len += snprintf(dmesgline + len, sizeof(dmesgline) - len,
310                                 " c=%d", props.batteryCurrent);
311             }
312
313             if (!mHealthdConfig->batteryFullChargePath.isEmpty()) {
314                 len += snprintf(dmesgline + len, sizeof(dmesgline) - len,
315                                 " fc=%d", props.batteryFullCharge);
316             }
317
318             if (!mHealthdConfig->batteryCycleCountPath.isEmpty()) {
319                 len += snprintf(dmesgline + len, sizeof(dmesgline) - len,
320                                 " cc=%d", props.batteryCycleCount);
321             }
322         } else {
323             len = snprintf(dmesgline, sizeof(dmesgline),
324                  "battery none");
325         }
326
327         snprintf(dmesgline + len, sizeof(dmesgline) - len, " chg=%s%s%s",
328                  props.chargerAcOnline ? "a" : "",
329                  props.chargerUsbOnline ? "u" : "",
330                  props.chargerWirelessOnline ? "w" : "");
```

FILE SYSTEMS FOR EXAMINATION

- rootfs — Where the kernel mounts the root file system, top of the tree
- tmpfs — Contain all files in virtual memory backed by RAM, Swap & Cache
- cgroup
- proc — Info on kernel, processes and configuration parameters
- sysfs — configuration and control files for device
- devpts

- **Fat**
- **Ext**
- **YAFFS2**

FILE SYSTEMS FOR EXAMINATION

- File system mount points to be examined
- Use adb shell mount to see the file systems on the mounted device

Mount Point	File System Type	Relevance
/proc	proc	Examine on the phone with the "cat" command. Look for relevant metadata about the system such as file system statistics
/data/data (on older systems, entire /data is 1 partition/file system)	YAFFS2	Nearly all app data
/data (on newer phones /data can be further segemented)	EXT3/EXT4/YAFFS2	App and system data excluding the app data stores found in /data/data
/cache	YAFFS2/EXT3	Cache file system used by some apps and by the system
/mnt/asec	tmpfs	Unencrypted app .apk file, which is stored encrypted on the SD card but decrypted here for running systems to access and utilize
/app-cache	tmpfs	Temporary file system where com.android.browser (on HTC Incredible) stores cache. Other apps may also use this directory over time
/mnt/sdcard	vfat	FAT32 file system on removable SD card
/mnt/emmc	vfat	FAT32 file system on the Embedded MultiMediaCard (eMMC)

FILESYSTEMS - FAT32

- In linux the filesystem driver is called VFAT
- Mount Points include
    - /mnt/sdcard
    - /mnt/secure/asec [encrypted partition]
    - /mnt/emmc

RAM

- Password
- Encryption keys
- Usernames
- App data
- Data from processes and services

- Identify a process id using $adb shell ps
  - $adb shell — start interactive shel
  - $su — gain root access
  - #chmod 777 /data/misc
  - $kill -10 pid — dump memory for process 10
  - $ls -l — look for file that is left
- Apply strings to the .hprof file

ACQUISITION

- ADB
- Logical
- Physical

ADB COMMANDS

- The adb devices command
- The adb push command
- The adb pull command
- The adb reboot command
- The adb reboot-bootloader and adb reboot recovery commands
- The fastboot devices command
- The fastboot oem unlock command
- The adb shell command
- The adb install command
- The adb logcat command

phone or emulator

Android SDK

IMAGING

- logical techniques
    - adb pull /data adbpull

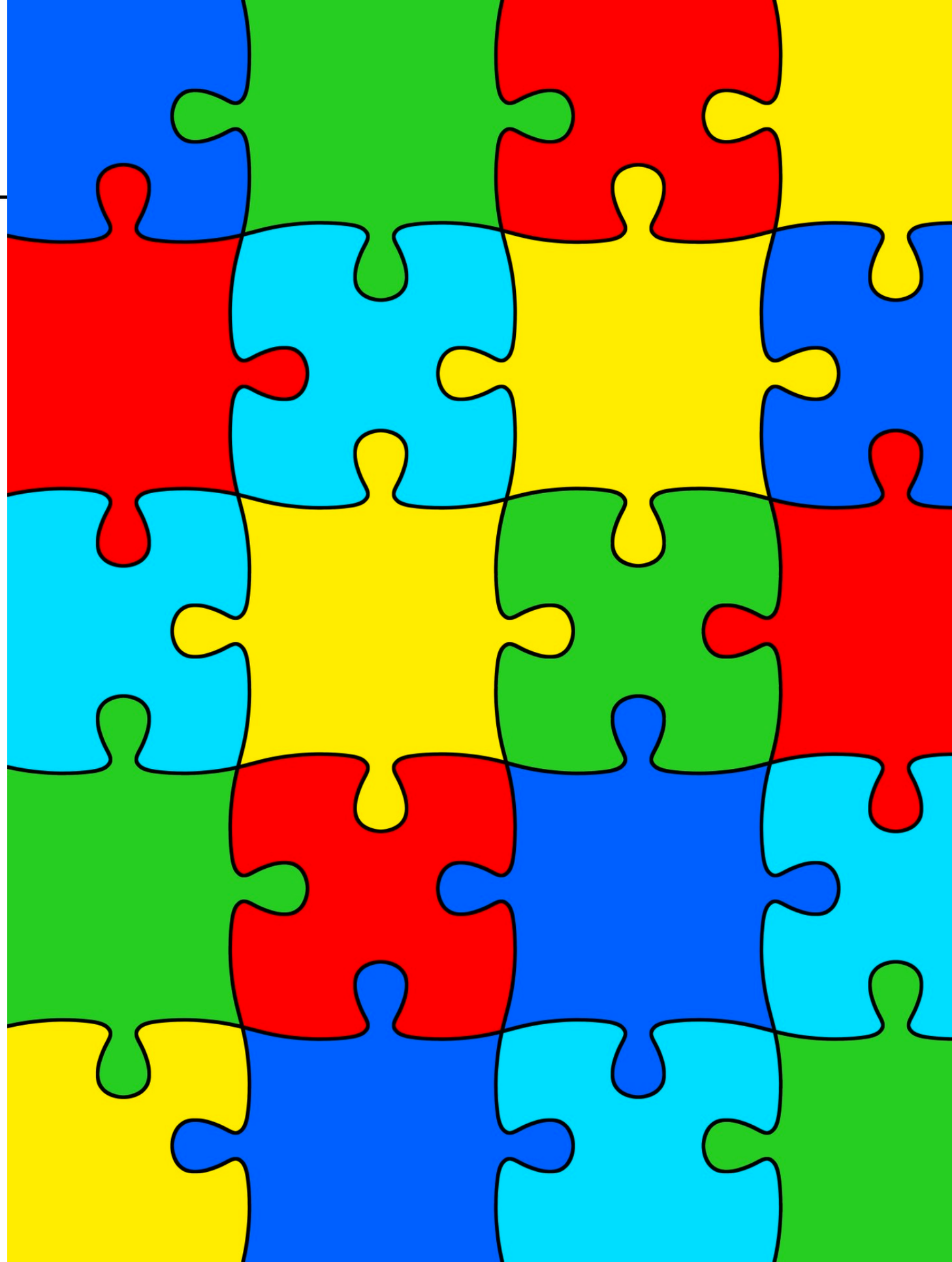
```
[redacted]tu:~$ adb pull /data adbpull/
pull: building file list...
<snip>
pull: /data/miscrild_nitz_long_name_31026 -> data/misc/rild_nitz_long_name_31026
pull: /data/misc/akmd_set.txt -> data/misc/akmd_set.txt

712 files pulled. 0 files skipped.
963 KB/s (208943249 bytes in 211.671s)
```

IMAGING

- physical techniques (AFLogical)
- Acquire root privileges on the device (vary by device)
- Identify the NAND flash partitions required
- Use a dump command to copy the data

ANALYSIS

- Techniques
- Sample application
- SQLite



ANALYSIS TECHNIQUES

- Timeline Analysis - log2timeline - Kristinn Gudjonnson
    - http://log2timeline.net/
- File System Analysis - The Sleuth Kit - Brian Carrier
    - http://www.sleuthkit.org/
- File Carving - scalpel -
- Strings
- Hexeditor
- Bulk Extractor

```
/
├── app-cache
│   └── com.android.browser
│       └── cache
│           └── webviewCache
├── cache
│   ├── lost+found
│   └── recovery
├── data
│   ├── anr
│   ├── app
│   ├── app-private
│   ├── backup
│   ├── btips
│   ├── dalvik-cache
│   ├── data
│   │   ├── com.facebook.katana
│   │   │   ├── cache
│   │   │   │   └── webviewCache
│   │   │   ├── databases
│   │   │   ├── files
│   │   │   ├── lib
│   │   │   └── shared_prefs
│   ├── dontpanic
│   ├── local
│   ├── lost+found
│   ├── misc
│   │   ├── bluetooth
│   │   ├── bluetoothd
│   │   ├── dhcp
│   │   ├── keystore
│   │   ├── lockscreen
│   │   ├── systemkeys
│   │   ├── vpn
│   │   └── wifi
│   ├── property
│   ├── system
│   │   ├── registered_services
│   │   ├── shared_prefs
│   │   ├── sync
│   │   ├── throttle
│   │   └── usagestats
│   └── tombstones
├── mnt
│   ├── asec
│   ├── emmc
│   │   ├── Android
│   │   │   └── data
│   │   │       └── com.android.providers.media
│   │   │           └── albumthumbs
│   │   ├── DCIM
│   │   │   └── 100MEDIA
│   │   ├── LOST.DIR
│   │   ├── MP3
│   │   │   ├── People Under the Stairs
│   ├── sdcard
│   │   ├── Android
│   │   │   └── data
│   │   │       ├── com.google.android.apps.maps
│   │   │       │   ├── cache
│   │   │       │   ├── debug
│   │   │       │   └── testdata
│   │   │       └── com.yelp.android
│   │   │           └── cache
│   │   │               └── images
│   │   ├── dcim
│   │   ├── download
│   │   ├── Downloads
│   │   ├── LOST.DIR
│   │   └── tmp
│   └── secure
│       ├── asec
│       └── staging
└── system
    ├── app
    ├── bin
    ├── customize
    │   ├── CID
    │   ├── MNS
    │   └── resource
    ├── etc
    │   ├── bluetooth
    │   ├── clockwidget
    │   ├── dhcpcd
    │   │   └── dhcpcd-hooks
    │   ├── firmware
    │   ├── iproute2
    │   ├── permissions
    │   ├── ppp
    │   ├── security
    │   ├── updatecmds
    │   ├── wifi
    │   └── wimax
    ├── fonts
    ├── framework
    ├── lib
    │   ├── bluez-plugin
    │   ├── egl
    │   ├── hw
    │   │   └── modules
    ├── lost+found
    ├── media
    │   └── audio
    │       ├── alarms
    │       ├── notifications
    │       ├── ringtones
    │       └── ui
    ├── tts
    │   └── lang_pico
    ├── usr
    │   ├── keychars
    │   ├── keylayout
    │   ├── share
    │   │   ├── bmd
    │   │   └── zoneinfo
    │   └── srec
    └── xbin
```

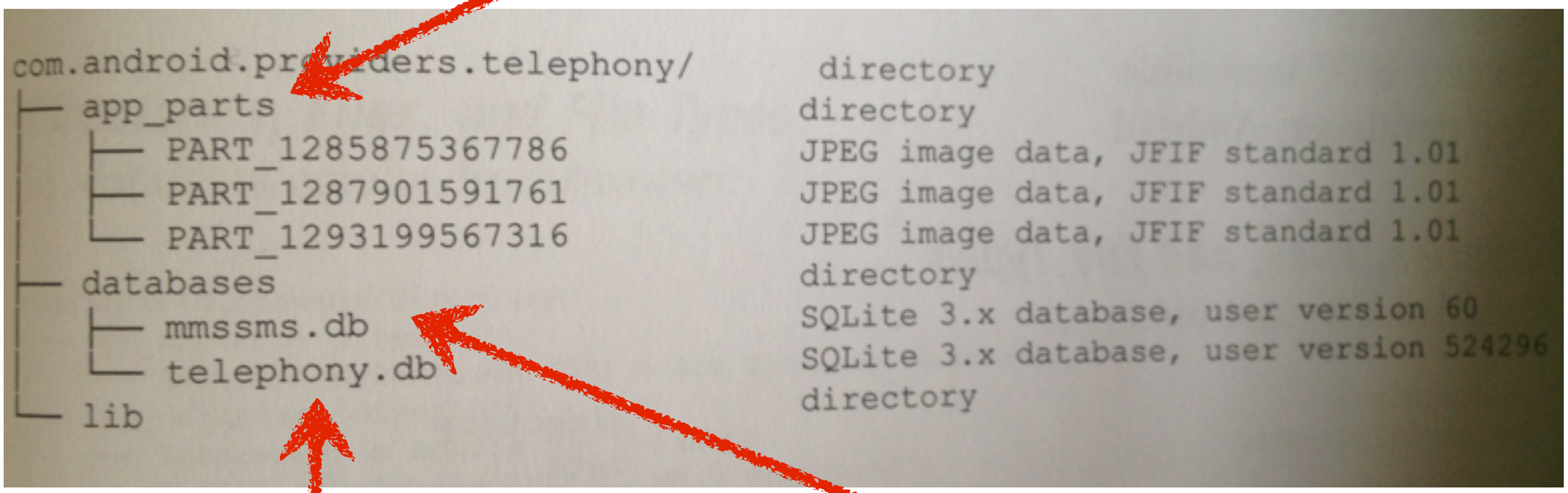
```
/
├── app-cache
│   └── com.android.browser
│       └── cache
│           └── webviewCache
├── cache
│   ├── lost+found
│   └── recovery
├── data
│   ├── anr
│   ├── app
│   ├── app-private
│   ├── backup
│   ├── btips
│   ├── dalvik-cache
│   ├── data
│   │   ├── com.facebook.katana
│   │   │   ├── cache
│   │   │   │   └── webviewCache
│   │   │   ├── databases
│   │   │   ├── files
│   │   │   ├── lib
│   │   │   └── shared_prefs
│   ├── dontpanic
│   ├── local
│   ├── lost+found
│   ├── misc
│   │   ├── bluetooth
│   │   ├── bluetoothd
│   │   ├── dhcp
│   │   ├── keystore
│   │   ├── lockscreen
│   │   ├── systemkeys
│   │   ├── vpn
│   │   └── wifi
│   ├── property
│   ├── system
│   │   ├── registered_services
│   │   ├── shared_prefs
│   │   ├── sync
│   │   ├── throttle
│   │   └── usagestats
│   └── tombstones
├── mnt
│   ├── asec
│   ├── emmc
│   │   ├── Android
│   │   │   └── data
│   │   │       └── com.android.providers.media
│   │   │           └── albumthumbs
│   │   ├── DCIM
│   │   │   └── 100MEDIA
│   │   ├── LOST.DIR
│   │   ├── MP3
│   │   │   ├── People Under the Stairs
├── sdcard
│   ├── Android
│   │   └── data
│   │       ├── com.google.android.apps.maps
│   │       │   ├── cache
│   │       │   ├── debug
│   │       │   ├── testdata
│   │       └── com.yelp.android
│   │           └── cache
│   │               └── images
│   ├── dcim
│   ├── download
│   ├── Downloads
│   ├── LOST.DIR
│   └── tmp
│   └── secure
│       ├── asec
│       └── staging
└── system
    ├── app
    ├── bin
    ├── customize
    │   ├── CID
    │   ├── MNS
    │   └── resource
    ├── etc
    │   ├── bluetooth
    │   ├── clockwidget
    │   ├── dhcpcd
    │   │   └── dhcpcd-hooks
    │   ├── firmware
    │   ├── iproute2
    │   ├── permissions
    │   ├── ppp
    │   ├── security
    │   ├── updatecmds
    │   ├── wifi
    │   └── wimax
    ├── fonts
    ├── framework
    ├── lib
    │   ├── bluez-plugin
    │   ├── egl
    │   ├── hw
    │   │   └── modules
    ├── lost+found
    ├── media
    │   └── audio
    │       ├── alarms
    │       ├── notifications
    │       ├── ringtones
    │       └── ui
    ├── tts
    │   └── lang_pico
    ├── usr
    │   ├── keychars
    │   ├── keylayout
    │   ├── share
    │   │   ├── bmd
    │   │   └── zoneinfo
    │   └── srec
    └── xbin
```

MESSAGING SMS & MMS

- The default app shipped with Android that handles SMS and MMS
- App Name: Messaging
- Package Name: com.android.providers.telephony
- Device: HTC
- App Developed: Android

Database Tables/Files	Description
words_content	Content of SMS Message
part	MMS attachment details such as type, name, location of file on system
sms	Full SMS message including to from and contents

app_parts folder contains file attachments. Files types are not maintained so file signature tools should be used



contains service information for the wireless carrier

sms table contains all the mesages and should be the primary focus

SUMMARY

- Boot process
- Storage
- Acquisition
- Analysis