ADB 命令大全

目　录

1. 基本用法 5

命令语法 5

为命令指定目标设备 5

启动/停止 6

查看 adb 版本 6

以 root 权限运行 adbd 7

指定 adb server 的网络端口 8

2. 设备连接管理 8

查询已连接设备/模拟器 8

USB 连接 9

无线连接 10

3. 应用管理 12

查看应用列表 12

安装 APK 14

卸载应用 22

清除应用数据与缓存 22

查看前台 Activity 23

4. 与应用交互 23
调起 Activity 25
调起 Service 25
发送广播 26
强制停止应用 26
5. 文件管理 27
复制设备里的文件到电脑 27
复制电脑里的文件到设备 27
6. 模拟按键/输入 28
电源键 31
菜单键 31
HOME 键 32
返回键 32
音量控制 32
媒体控制 32
点亮/熄灭屏幕 33
滑动解锁 34
输入文本 34
7. 查看日志 34
Android 日志 34
内核日志 38
8. 查看设备信息 39

型号 39
电池状况 40
屏幕分辨率 40
屏幕密度 41
显示屏参数 41
android_id 41
IMEI 42
Android 系统版本 43
Mac 地址 43
CPU 信息 43
更多硬件与系统属性 44
9. 实用功能 46
屏幕截图 46
录制屏幕 46
重新挂载 system 分区为可写 47
查看连接过的 WiFi 密码 50
设置系统日期和时间 51
重启手机 51
检测设备是否已 root 51
使用 Monkey 进行压力测试 52
10. 刷机相关命令 52
重启到 Recovery 模式 52

从 Recovery 重启到 Android 52
重启到 Fastboot 模式 53
通过 sideload 更新系统 53
11. 更多 adb shell 命令 54
查看进程 54
查看实时资源占用情况 55
其它 57

1. 基本用法

命令语法

adb 命令的基本语法如下：

```
adb [-d|-e|-s <serialNumber>] <command>
```

如果只有一个设备/模拟器连接时，可以省略掉 `[-d|-e|-s <serialNumber>]` 这一部分，直接使用 `adb <command>`。

为命令指定目标设备

如果有多个设备/模拟器连接，则需要为命令指定目标设备。

参数	含义
-d	指定当前唯一通过 USB 连接的 Android 设备为命令目标
-e	指定当前唯一运行的模拟器为命令目标
`-s <serialNumber>`	指定相应 serialNumber 号的设备/模拟器为命令目标

在多个设备/模拟器连接的情况下较常用的是 `-s <serialNumber>` 参数，serialNumber 可以通过 `adb devices` 命令获取。如：

```
$ adb devices
List of devices attached
```

```
cf264b8f    device
emulator-5554   device
```

输出里的 `cf264b8f` 和 `emulator-5554` 即为 serialNumber。比如这时想指定 `cf264b8f` 这个设备来运行 adb 命令获取屏幕分辨率：

```
adb -s cf264b8f shell wm size
```

遇到多设备/模拟器的情况均使用这几个参数为命令指定目标设备，下文中为简化描述，不再重复。

启动/停止

启动 adb server 命令：

```
adb start-server
```

（一般无需手动执行此命令，在运行 adb 命令时若发现 adb server 没有启动会自动调起。）

停止 adb server 命令：

```
adb kill-server
```

查看 adb 版本

命令：

```
adb version
```

示例输出：

```
Android Debug Bridge version 1.0.32
```

```
Revision 09a0d98bebce-android
```

以 root 权限运行 adbd

adb 的运行原理是 PC 端的 adb server 与手机端的守护进程 adbd 建立连接，然后 PC 端的 adb client 通过 adb server 转发命令，adbd 接收命令后解析运行。

所以如果 adbd 以普通权限执行，有些需要 root 权限才能执行的命令无法直接用 `adb xxx` 执行。这时可以 `adb shell` 然后 `su` 后执行命令，也可以让 adbd 以 root 权限执行，这个就能随意执行高权限命令了。

命令：

```
adb root
```

正常输出：

```
restarting adbd as root
```

现在再运行 `adb shell`，看看命令行提示符是不是变成 `#`了？

有些手机 root 后也无法通过 `adb root` 命令让 adbd 以 root 权限执行，比如三星的部分机型，会提示 `adbd cannot run as root in production builds`，此时可以先安装 adbd Insecure，然后 `adb root` 试试。

相应地，如果要恢复 adbd 为非 root 权限的话，可以使用 `adb unroot` 命令。

指定 adb server 的网络端口

命令：

```
adb -P <port> start-server
```

默认端口为 5037。

2. 设备连接管理

查询已连接设备/模拟器

命令：

```
adb devices
```

输出示例：

```
List of devices attached
cf264b8f    device
emulator-5554   device
```

输出格式为 `[serialNumber] [state]`，serialNumber 即我们常说的 SN，state 有如下几种：

- `offline` —— 表示设备未连接成功或无响应。

- `device` —— 设备已连接。注意这个状态并不能标识 Android 系统已经完全启动和可操作，在设备启动过程中设备实例就可连接到 adb，但启动完毕后系统才处于可操作状态。
- `no device` —— 没有设备/模拟器连接。

以上输出显示当前已经连接了两台设备/模拟器，`cf264b8f` 与 `emulator-5554` 分别是它们的 SN。从 `emulator-5554` 这个名字可以看出它是一个 Android 模拟器。

常见异常输出：

1. 没有设备/模拟器连接成功。
2. `List of devices attached`
3. 设备/模拟器未连接到 adb 或无响应。
4. `List of devices attached`
5. `cf264b8f offline`

USB 连接

通过 USB 连接来正常使用 adb 需要保证几点：

1. 硬件状态正常。

   包括 Android 设备处于正常开机状态，USB 连接线和各种接口完好。

2. Android 设备的开发者选项和 USB 调试模式已开启。

可以到「设置」–「开发者选项」–「Android 调试」查看。

如果在设置里找不到开发者选项，那需要通过一个彩蛋来让它显示出来：在「设置」–「关于手机」连续点击「版本号」7 次。

3. 设备驱动状态正常。

   这一点貌似在 Linux 和 Mac OS X 下不用操心，在 Windows 下有可能遇到需要安装驱动的情况，确认这一点可以右键「计算机」–「属性」，到「设备管理器」里查看相关设备上是否有黄色感叹号或问号，如果没有就说明驱动状态已经好了。否则可以下载一个手机助手类程序来安装驱动先。

4. 通过 USB 线连接好电脑和设备后确认状态。
5. `adb devices`

   如果能看到

   `xxxxxx device`

   说明连接成功。

无线连接

除了可以通过 USB 连接设备与电脑来使用 adb，也可以通过无线连接——虽然连接过程中也有需要使用 USB 的步骤，但是连接成功之后你的设备就可以在一定范围内摆脱 USB 连接线的限制啦！

操作步骤：

1. 将 Android 设备与将运行 adb 的电脑连接到同一个局域网，比如连到同一个 WiFi。
2. 将设备与电脑通过 USB 线连接。

   应确保连接成功（可运行 `adb devices` 看是否能列出该设备）。

3. 让设备在 5555 端口监听 TCP/IP 连接：
4. `adb tcpip 5555`
5. 断开 USB 连接。
6. 找到设备的 IP 地址。

   一般能在「设置」–「关于手机」–「状态信息」–「IP 地址」找到。

7. 通过 IP 地址连接设备。
8. `adb connect <device-ip-address>`

   这里的 `<device-ip-address>` 就是上一步中找到的设备 IP 地址。

9. 确认连接状态。
10. `adb devices`

    如果能看到

    `<device-ip-address>:5555 device`

说明连接成功。

如果连接不了，请确认 Android 设备与电脑是连接到了同一个 WiFi，然后再次执行 `adb connect <device-ip-address>` 那一步；

如果还是不行的话，通过 `adb kill-server` 重新启动 adb 然后从头再来一次试试。

断开无线连接

命令：

```
adb disconnect <device-ip-address>
```

3. 应用管理

查看应用列表

查看应用列表的基本命令格式是

```
adb shell pm list packages [-f] [-d] [-e] [-s] [-3] [-i] [-u] [--user USER_ID] [FILTER]
```

即在 `adb shell pm list packages` 的基础上可以加一些参数进行过滤查看不同的列表，支持的过滤参数如下：

参数	显示列表
无	所有应用

参数	显示列表
–f	显示应用关联的 apk 文件
–d	只显示 disabled 的应用
–e	只显示 enabled 的应用
–s	只显示系统应用
–3	只显示第三方应用
–i	显示应用的 installer
–u	包含已卸载应用
`<FILTER>`	包名包含 `<FILTER>` 字符串

所有应用

命令：

```
adb shell pm list packages
```

输出示例：

```
package:com.android.smoketest
package:com.example.android.livecubes
package:com.android.providers.telephony
package:com.google.android.googlequicksearchbox
package:com.android.providers.calendar
package:com.android.providers.media
package:com.android.protips
```

```
package:com.android.documentsui
package:com.android.gallery
package:com.android.externalstorage
...
// other packages here
...
```

系统应用

命令：

```
adb shell pm list packages -s
```

第三方应用

命令：

```
adb shell pm list packages -3
```

包名包含某字符串的应用

比如要查看包名包含字符串 `mazhuang` 的应用列表，命令：

```
adb shell pm list packages mazhuang
```

当然也可以使用 grep 来过滤：

```
adb shell pm list packages | grep mazhuang
```

安装 APK

命令：

```
adb install <apk file>
```

参数：

`adb install` 后面可以跟一些参数来控制安装 APK 的行为，常用参数及含义如下：

参数	含义
–r	允许覆盖安装。
–s	将应用安装到 sdcard。
–d	允许降级覆盖安装。

完整参数列表及含义可以直接运行 `adb` 命令然后查看 `adb install [-lrtsdg] <file>` 一节。

如果见到类似如下输出（状态为 `Success`）代表安装成功：

```
12040 KB/s (22205609 bytes in 1.801s)
        pkg: /data/local/tmp/SogouInput_android_v8.3_sweb.apk
Success
```

而如果状态为 `Failure` 则表示安装失败。常见安装失败输出代码、含义及可能的解决办法如下：

输出	含义	解决办法
INSTALL_FAILED_ALREADY_EXISTS	应用已经存在	使用 `-r` 参数

输出	含义	解决办法
INSTALL_FAILED_INVALID_APK	无效的 APK 文件	
INSTALL_FAILED_INVALID_URI	无效的 APK 文件名	确保 APK 文件名里无中文
INSTALL_FAILED_INSUFFICIENT_STORAGE	空间不足	清理空间
INSTALL_FAILED_DUPLICATE_PACKAGE	已经存在同名程序	
INSTALL_FAILED_NO_SHARED_USER	请求的共享用户不存在	
INSTALL_FAILED_UPDATE_INCOMPATIBLE	已经安装过签名不一样的同名应用，且数据没有移除	
INSTALL_FAILED_SHARED_USER_INCOMPATIBLE	请求的共享用户存在但签名不一致	
INSTALL_FAILED_MISSING_SHARED_LIBRARY	安装包使用了设备上不可用的共享库	
INSTALL_FAILED_REPLACE_COULDNT_DELETE	替换时无法删除	
INSTALL_FAILED_DEXOPT	dex 优化验证失败或空间不足	

输出	含义	解决办法
INSTALL_FAILED_OLDER_SDK	设备系统版本低于应用要求	
INSTALL_FAILED_CONFLICTING_PROVIDER	设备里已经存在与应用里同名的 content provider	
INSTALL_FAILED_NEWER_SDK	设备系统版本高于应用要求	
INSTALL_FAILED_TEST_ONLY	应用是 test-only 的，但安装时没有指定 `-t` 参数	
INSTALL_FAILED_CPU_ABI_INCOMPATIBLE	包含不兼容设备 CPU 应用程序二进制接口的 native code	
INSTALL_FAILED_MISSING_FEATURE	应用使用了设备不可用的功能	
INSTALL_FAILED_CONTAINER_ERROR	sdcard 访问失败	确认 sdcard 可用，或者安装到内置存储
INSTALL_FAILED_INVALID_INSTALL_LOCATION	不能安装到指定位置	切换安装位置，添加或删

输出	含义	解决办法
		除 `-s` 参数
INSTALL_FAILED_MEDIA_UNAVAILABLE	安装位置不可用	一般为 sdcard，确认 sdcard 可用或安装到内置存储
INSTALL_FAILED_VERIFICATION_TIMEOUT	验证安装包超时	
INSTALL_FAILED_VERIFICATION_FAILURE	验证安装包失败	
INSTALL_FAILED_PACKAGE_CHANGED	应用与调用程序期望的不一致	
INSTALL_FAILED_UID_CHANGED	以前安装过该应用，与本次分配的 UID 不一致	清除以前安装过的残留文件
INSTALL_FAILED_VERSION_DOWNGRADE	已经安装了该应用更高版本	使用 `-d` 参数

输出	含义	解决办法
INSTALL_FAILED_PERMISSION_MODEL_DOWNGRADE	已安装 target SDK 支持运行时权限的同名应用，要安装的版本不支持运行时权限	
INSTALL_PARSE_FAILED_NOT_APK	指定路径不是文件，或不是以`.apk`结尾	
INSTALL_PARSE_FAILED_BAD_MANIFEST	无法解析的 AndroidManifest.xml 文件	
INSTALL_PARSE_FAILED_UNEXPECTED_EXCEPTION	解析器遇到异常	
INSTALL_PARSE_FAILED_NO_CERTIFICATES	安装包没有签名	
INSTALL_PARSE_FAILED_INCONSISTENT_CERTIFICATES	已安装该应用，且签名与 APK 文件不一致	先卸载设备上的该应用，再安装
INSTALL_PARSE_FAILED_CERTIFICATE_ENCODING	解析 APK 文件时遇到`CertificateEncodingException`	
INSTALL_PARSE_FAILED_BAD_PACKAGE_NAME	manifest 文件里没有或者使用了无效的包名	
INSTALL_PARSE_FAILED_BAD_SHARED_USER_ID	manifest 文件里指定了无效的	

输出	含义	解决办法
	共享用户 ID	
INSTALL_PARSE_FAILED_MANIFEST_MALFORMED	解析 manifest 文件时遇到结构性错误	
INSTALL_PARSE_FAILED_MANIFEST_EMPTY	在 manifest 文件里找不到找可操作标签（instrumentation 或 application）	
INSTALL_FAILED_INTERNAL_ERROR	因系统问题安装失败	
INSTALL_FAILED_USER_RESTRICTED	用户被限制安装应用	
INSTALL_FAILED_DUPLICATE_PERMISSION	应用尝试定义一个已经存在的权限名称	
INSTALL_FAILED_NO_MATCHING_ABIS	应用包含设备的应用程序二进制接口不支持的 native code	
INSTALL_CANCELED_BY_USER	应用安装需要在设备上确认，但未操作设备或点了取消	在设备上同意安装
INSTALL_FAILED_ACWF_INCOMPATIBLE	应用程序与设备不兼容	
does not contain AndroidManifest.xml	无效的 APK 文件	

输出	含义	解决办法
is not a valid zip file	无效的 APK 文件	
Offline	设备未连接成功	先将设备与 adb 连接成功
unauthorized	设备未授权允许调试	
error: device not found	没有连接成功的设备	先将设备与 adb 连接成功
protocol failure	设备已断开连接	先将设备与 adb 连接成功
Unknown option: -s	Android 2.2 以下不支持安装到 sdcard	不使用 `-s` 参数
No space left on devicerm	空间不足	清理空间
Permission denied ... sdcard ...	sdcard 不可用	

参考：PackageManager.java

卸载应用

命令：

```
adb uninstall [-k] <packagename>
```

`<packagename>` 表示应用的包名，`-k` 参数可选，表示卸载应用但保留数据和缓存目录。

命令示例：

```
adb uninstall com.qihoo360.mobilesafe
```

表示卸载 360 手机卫士。

清除应用数据与缓存

命令：

```
adb shell pm clear <packagename>
```

`<packagename>` 表示应用名包，这条命令的效果相当于在设置里的应用信息界面点击了「清除缓存」和「清除数据」。

命令示例：

```
adb shell pm clear com.qihoo360.mobilesafe
```

表示清除 360 手机卫士的数据和缓存。

查看前台 Activity

命令：

```
adb shell dumpsys activity activities | grep mFocusedActivity
```

输出示例：

```
mFocusedActivity: ActivityRecord{8079d7e u0 com.cyanogenmod.trebuchet/com.android.launcher3.Launcher t42}
```

其中的 `com.cyanogenmod.trebuchet/com.android.launcher3.Launcher` 就是当前处于前台的 Activity。

4. 与应用交互

主要是使用 `am <command>` 命令，常用的 `<command>` 如下：

command	用途
`start [options] <INTENT>`	启动 `<INTENT>` 指定的 Activity
`startservice [options] <INTENT>`	启动 `<INTENT>` 指定的 Service
`broadcast [options] <INTENT>`	发送 `<INTENT>` 指定的广播
`force-stop <packagename>`	停止 `<packagename>` 相关的进程

`<INTENT>` 参数很灵活，和写 Android 程序时代码里的 Intent 相对应。

用于决定 intent 对象的选项如下：

参数	含义
`-a <ACTION>`	指定 action，比如 `android.intent.action.VIEW`
`-c <CATEGORY>`	指定 category，比如 `android.intent.category.APP_CONTACTS`
`-n <COMPONENT>`	指定完整 component 名，用于明确指定启动哪个 Activity，如 `com.example.app/.ExampleActivity`

`<INTENT>` 里还能带数据，就像写代码时的 Bundle 一样：

参数	含义
`--esn <EXTRA_KEY>`	null 值（只有 key 名）
`-e|--es <EXTRA_KEY> <EXTRA_STRING_VALUE>`	string 值
`--ez <EXTRA_KEY> <EXTRA_BOOLEAN_VALUE>`	boolean 值
`--ei <EXTRA_KEY> <EXTRA_INT_VALUE>`	integer 值
`--el <EXTRA_KEY> <EXTRA_LONG_VALUE>`	long 值
`--ef <EXTRA_KEY> <EXTRA_FLOAT_VALUE>`	float 值

参数	含义
`--eu <EXTRA_KEY> <EXTRA_URI_VALUE>`	URI
`--ecn <EXTRA_KEY> <EXTRA_COMPONENT_NAME_VALUE>`	component name
`--eia <EXTRA_KEY> <EXTRA_INT_VALUE>[,<EXTRA_INT_VALUE...]`	integer 数组
`--ela <EXTRA_KEY> <EXTRA_LONG_VALUE>[,<EXTRA_LONG_VALUE...]`	long 数组

调起 Activity

命令格式：

```
adb shell am start [options] <INTENT>
```

例如：

```
adb shell am start -n com.tencent.mm/.ui.LauncherUI
```

表示调起微信主界面。

```
adb shell am start -n
org.mazhuang.boottimemeasure/.MainActivity --es "toast"
"hello, world"
```

表示调起 `org.mazhuang.boottimemeasure/.MainActivity` 并传给它 string 数据键值对 `toast - hello, world`。

调起 Service

命令格式：

```
adb shell am startservice [options] <INTENT>
```

例如：

```
adb shell am startservice -n
com.tencent.mm/.plugin.accountsync.model.AccountAuthenticat
orService
```

表示调起微信的某 Service。

发送广播

命令格式：

```
adb shell am broadcast [options] <INTENT>
```

例如：

```
adb shell am broadcast -a
android.intent.action.BOOT_COMPLETED -n
org.mazhuang.boottimemeasure/.BootCompletedReceiver
```

表示向 `org.mazhuang.boottimemeasure/.BootCompletedReceiver` 发送一个 `BOOT_COMPLETED` 广播，这类用法在测试的时候很实用，比如某个广播的场景很难制造，可以考虑通过这种方式来发送广播。

强制停止应用

命令：

```
adb shell am force-stop <packagename>
```

命令示例：

```
adb shell am force-stop com.qihoo360.mobilesafe
```

表示停止 360 安全卫士的一切进程与服务。

5. 文件管理

复制设备里的文件到电脑

命令：

```
adb pull <设备里的文件路径> [电脑上的目录]
```

其中 `电脑上的目录` 参数可以省略，默认复制到当前目录。

例：

```
adb pull /sdcard/sr.mp4 ~/tmp/
```

*小技巧：*设备上的文件路径可能需要 root 权限才能访问，如果你的设备已经 root 过，可以先使用 `adb shell` 和 `su` 命令在 adb shell 里获取 root 权限后，先 `cp /path/on/device /sdcard/filename` 将文件复制到 sdcard，然后 `adb pull /sdcard/filename /path/on/pc`。

复制电脑里的文件到设备

命令：

```
adb push <电脑上的文件路径> <设备里的目录>
```

例：

```
adb push ~/sr.mp4 /sdcard/
```

*小技巧：*设备上的文件路径普通权限可能无法直接写入，如果你的设备已经 root 过，可以先 `adb push /path/on/pc /sdcard/filename`，然后 `adb shell` 和 `su` 在 adb shell 里获取 root 权限后，`cp /sdcard/filename /path/on/device`。

6. 模拟按键/输入

在 `adb shell` 里有个很实用的命令叫 `input`，通过它可以做一些有趣的事情。

`input` 命令的完整 help 信息如下：

```
Usage: input [<source>] <command> [<arg>...]

The sources are:
      mouse
      keyboard
      joystick
      touchnavigation
      touchpad
      trackball
      stylus
      dpad
      gesture
      touchscreen
      gamepad

The commands and default sources are:
      text <string> (Default: touchscreen)
```

```
    keyevent [--longpress] <key code number or name> ... (Default: keyboard)
    tap <x> <y> (Default: touchscreen)
    swipe <x1> <y1> <x2> <y2> [duration(ms)] (Default: touchscreen)
    press (Default: trackball)
    roll <dx> <dy> (Default: trackball)
```

比如使用 `adb shell input keyevent <keycode>` 命令，不同的 keycode 能实现不同的功能，完整的 keycode 列表详见 KeyEvent，摘引部分我觉得有意思的如下：

keycode	含义
3	HOME 键
4	返回键
5	打开拨号应用
6	挂断电话
24	增加音量
25	降低音量
26	电源键
27	拍照（需要在相机应用里）
64	打开浏览器

keycode	含义
82	菜单键
85	播放/暂停
86	停止播放
87	播放下一首
88	播放上一首
122	移动光标到行首或列表顶部
123	移动光标到行末或列表底部
126	恢复播放
127	暂停播放
164	静音
176	打开系统设置
187	切换应用
207	打开联系人
208	打开日历
209	打开音乐

keycode	含义
210	打开计算器
220	降低屏幕亮度
221	提高屏幕亮度
223	系统休眠
224	点亮屏幕
231	打开语音助手
276	如果没有 wakelock 则让系统休眠

下面是 `input` 命令的一些用法举例。

电源键

命令：

```
adb shell input keyevent 26
```

执行效果相当于按电源键。

菜单键

命令：

```
adb shell input keyevent 82
```

HOME 键

命令：

```
adb shell input keyevent 3
```

返回键

命令：

```
adb shell input keyevent 4
```

音量控制

增加音量：

```
adb shell input keyevent 24
```

降低音量：

```
adb shell input keyevent 25
```

静音：

```
adb shell input keyevent 164
```

媒体控制

播放/暂停：

```
adb shell input keyevent 85
```

停止播放：

```
adb shell input keyevent 86
```

播放下一首：

```
adb shell input keyevent 87
```

播放上一首：

```
adb shell input keyevent 88
```

恢复播放：

```
adb shell input keyevent 126
```

暂停播放：

```
adb shell input keyevent 127
```

点亮/熄灭屏幕

可以通过上文讲述过的模拟电源键来切换点亮和熄灭屏幕，但如果明确地想要点亮或者熄灭屏幕，那可以使用如下方法。

点亮屏幕：

```
adb shell input keyevent 224
```

熄灭屏幕：

```
adb shell input keyevent 223
```

滑动解锁

如果锁屏没有密码，是通过滑动手势解锁，那么可以通过 `input swipe` 来解锁。

命令（参数以机型 Nexus 5，向上滑动手势解锁举例）：

```
adb shell input swipe 300 1000 300 500
```

参数 `300 1000 300 500` 分别表示`起始点 x 坐标 起始点 y 坐标 结束点 x 坐标 结束点 y 坐标`。

输入文本

在焦点处于某文本框时，可以通过 `input` 命令来输入文本。

命令：

```
adb shell input text hello
```

现在 `hello` 出现在文本框了。

7. 查看日志

Android 系统的日志分为两部分，底层的 Linux 内核日志输出到 /proc/kmsg，Android 的日志输出到 /dev/log。

Android 日志

命令格式：

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

常用用法列举如下：

按级别过滤日志

Android 的日志分为如下几个级别：

- V —— Verbose（最低，输出得最多）
- D —— Debug
- I —— Info
- W —— Warning
- E —— Error
- F —— Fatal
- S —— Silent（最高，啥也不输出）

按某级别过滤日志则会将该级别及以上的日志输出。

比如，命令：

```
adb logcat *:W
```

会将 Warning、Error、Fatal 和 Silent 日志输出。

按 tag 和级别过滤日志

比如，命令：

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

表示输出 tag `ActivityManager` 的 Info 以上级别日志，输出 tag `MyApp` 的 Debug 以上级别日志，及其它 tag 的 Silent 级别日志（即屏蔽其它 tag 日志）。

日志格式

可以用 `adb logcat -v <format>` 选项指定日志输出格式。

日志支持按以下几种 `<format>`：

- brief

  默认格式。格式为：

  ```
  <priority>/<tag>(<pid>): <message>
  ```

  示例：

  ```
  D/HeadsetStateMachine( 1785): Disconnected process message: 10, size: 0
  ```

- process

  格式为：

  ```
  <priority>(<pid>) <message>
  ```

  示例：

  ```
  D( 1785) Disconnected process message: 10, size: 0 (HeadsetStateMachine)
  ```

- tag

  格式为：

  ```
  <priority>/<tag>: <message>
  ```

  示例：

  ```
  D/HeadsetStateMachine: Disconnected process message: 10, size: 0
  ```

- raw

  格式为：

  ```
  <message>
  ```

  示例：

  ```
  Disconnected process message: 10, size: 0
  ```

- time

  格式为：

  ```
  <datetime> <priority>/<tag>(<pid>): <message>
  ```

  示例：

  ```
  08-28 22:39:39.974 D/HeadsetStateMachine( 1785): Disconnected process message: 10, size: 0
  ```

- threadtime

格式为：

```
<datetime> <pid> <tid> <priority> <tag>: <message>
```

示例：

```
08-28 22:39:39.974  1785  1832 D HeadsetStateMachine:
Disconnected process message: 10, size: 0
```

- long

格式为：

```
[ <datetime> <pid>:<tid> <priority>/<tag> ]
<message>
```

示例：

```
[ 08-28 22:39:39.974  1785: 1832
D/HeadsetStateMachine ]
Disconnected process message: 10, size: 0
```

指定格式可与上面的过滤同时使用。比如：

```
adb logcat -v long ActivityManager:I *:S
```

清空日志

```
adb logcat -c
```

内核日志

命令：

```
adb shell dmesg
```

输出示例：

```
<6>[14201.684016] PM: noirq resume of devices complete after 0.982 msecs
<6>[14201.685525] PM: early resume of devices complete after 0.838 msecs
<6>[14201.753642] PM: resume of devices complete after 68.106 msecs
<4>[14201.755954] Restarting tasks ... done.
<6>[14201.771229] PM: suspend exit 2016-08-28 13:31:32.679217193 UTC
<6>[14201.872373] PM: suspend entry 2016-08-28 13:31:32.780363596 UTC
<6>[14201.872498] PM: Syncing filesystems ... done.
```

中括号里的 `[14201.684016]` 代表内核开始启动后的时间，单位为秒。

通过内核日志我们可以做一些事情，比如衡量内核启动时间，在系统启动完毕后的内核日志里找到 `Freeing init memory` 那一行前面的时间就是。

8. 查看设备信息

型号

命令：

```
adb shell getprop ro.product.model
```

输出示例：

```
Nexus 5
```

电池状况

命令：

```
adb shell dumpsys battery
```

输入示例：

```
Current Battery Service state:
  AC powered: false
  USB powered: true
  Wireless powered: false
  status: 2
  health: 2
  present: true
  level: 44
  scale: 100
  voltage: 3872
  temperature: 280
  technology: Li-poly
```

其中 `scale` 代表最大电量，`level` 代表当前电量。上面的输出表示还剩下 44% 的电量。

屏幕分辨率

命令：

```
adb shell wm size
```

输出示例：

```
Physical size: 1080x1920
```

该设备屏幕分辨率为 1080px * 1920px。

屏幕密度

命令：

```
adb shell wm density
```

输出示例：

```
Physical density: 420
```

该设备屏幕密度为 420dpi。

显示屏参数

命令：

```
adb shell dumpsys window displays
```

输出示例：

```
WINDOW MANAGER DISPLAY CONTENTS (dumpsys window displays)
  Display: mDisplayId=0
    init=1080x1920 420dpi cur=1080x1920 app=1080x1794
rng=1080x1017-1810x1731
    deferred=false layoutNeeded=false
```

其中 `mDisplayId` 为 显示屏编号，`init` 是初始分辨率和屏幕密度，`app` 的高度比 `init` 里的要小，表示屏幕底部有虚拟按键，高度为 1920 – 1794 = 126px 合 42dp。

android_id

命令：

```
adb shell settings get secure android_id
```

输出示例：

```
51b6be48bac8c569
```

IMEI

在 Android 4.4 及以下版本可通过如下命令获取 IMEI：

```
adb shell dumpsys iphonesubinfo
```

输出示例：

```
Phone Subscriber Info:
  Phone Type = GSM
  Device ID = 860955027785041
```

其中的 `Device ID` 就是 IMEI。

而在 Android 5.0 及以上版本里这个命令输出为空，得通过其它方式获取了（需要 root 权限）：

```
adb shell
su
service call iphonesubinfo 1
```

输出示例：

```
Result: Parcel(
  0x00000000: 00000000 0000000f 00360038 00390030 '........8.6.0.9.'
  0x00000010: 00350035 00320030 00370037 00350038 '5.5.0.2.7.7.8.5.'
  0x00000020: 00340030 00000031                   '0.4.1...       ')
```

把里面的有效内容提取出来就是 IMEI 了，比如这里的是 `860955027785041`。

参考：adb shell dumpsys iphonesubinfo not working since Android 5.0 Lollipop

Android 系统版本

命令：

```
adb shell getprop ro.build.version.release
```

输出示例：

```
5.0.2
```

Mac 地址

命令：

```
adb shell cat /sys/class/net/wlan0/address
```

输出示例：

```
f8:a9:d0:17:42:4d
```

CPU 信息

命令：

```
adb shell cat /proc/cpuinfo
```

输出示例：

```
Processor       : ARMv7 Processor rev 0 (v7l)
processor       : 0
BogoMIPS        : 38.40

processor       : 1
BogoMIPS        : 38.40

processor       : 2
BogoMIPS        : 38.40

processor       : 3
BogoMIPS        : 38.40

Features        : swp half thumb fastmult vfp edsp neon vfpv3
tls vfpv4 idiva idivt
CPU implementer : 0x51
CPU architecture: 7
CPU variant     : 0x2
CPU part        : 0x06f
CPU revision    : 0

Hardware        : Qualcomm MSM 8974 HAMMERHEAD (Flattened
Device Tree)
Revision        : 000b
Serial          : 0000000000000000
```

这是 Nexus 5 的 CPU 信息，我们从输出里可以看到使用的硬件是 `Qualcomm MSM 8974`，processor 的编号是 0 到 3，所以它是四核的，采用的架构是 `ARMv7 Processor rev 0 (v71)`。

更多硬件与系统属性

设备的更多硬件与系统属性可以通过如下命令查看：

```
adb shell cat /system/build.prop
```

这会输出很多信息，包括前面几个小节提到的「型号」和「Android 系统版本」等。

输出里还包括一些其它有用的信息，它们也可通过 `adb shell getprop <属性名>` 命令单独查看，列举一部分属性如下：

属性名	含义
ro.build.version.sdk	SDK 版本
ro.build.version.release	Android 系统版本
ro.build.version.security_patch	Android 安全补丁程序级别
ro.product.model	型号
ro.product.brand	品牌
ro.product.name	设备名
ro.product.board	处理器型号
ro.product.cpu.abilist	CPU 支持的 abi 列表
persist.sys.isUsbOtgEnabled	是否支持 OTG
dalvik.vm.heapsize	每个应用程序的内存上限
ro.sf.lcd_density	屏幕密度

9. 实用功能

屏幕截图

命令：

```
adb shell screencap -p /sdcard/sc.png
```

然后将 png 文件导出到电脑：

```
adb pull /sdcard/sc.png
```

可以使用 `adb shell screencap -h` 查看 `screencap` 命令的帮助信息，下面是两个有意义的参数及含义：

参数	含义
-p	指定保存文件为 png 格式
-d display-id	指定截图的显示屏编号（有多显示屏的情况下）

实测如果指定文件名以 `.png` 结尾时可以省略 -p 参数；否则需要使用 -p 参数。如果不指定文件名，截图文件的内容将直接输出到 stdout。

录制屏幕

录制屏幕以 mp4 格式保存到 /sdcard：

```
adb shell screenrecord /sdcard/filename.mp4
```

需要停止时按 Ctrl-C，默认录制时间和最长录制时间都是 180 秒。

如果需要导出到电脑：

```
adb pull /sdcard/filename.mp4
```

可以使用 `adb shell screenrecord --help` 查看 `screenrecord` 命令的帮助信息，下面是常见参数及含义：

参数	含义
--size WIDTHxHEIGHT	视频的尺寸，比如 `1280x720`，默认是屏幕分辨率。
--bit-rate RATE	视频的比特率，默认是 4Mbps。
--time-limit TIME	录制时长，单位秒。
--verbose	输出更多信息。

重新挂载 system 分区为可写

注：需要 root 权限。

/system 分区默认挂载为只读，但有些操作比如给 Android 系统添加命令、删除自带应用等需要对 /system 进行写操作，所以需要重新挂载它为可读写。

步骤：

1. 进入 shell 并切换到 root 用户权限。

   命令：

   ```
   adb shell
   su
   ```

2. 查看当前分区挂载情况。

   命令：

   ```
   mount
   ```

   输出示例：

   ```
   rootfs / rootfs ro,relatime 0 0
   tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,mode=755
   0 0
   devpts /dev/pts devpts rw,seclabel,relatime,mode=600 0
   0
   proc /proc proc rw,relatime 0 0
   sysfs /sys sysfs rw,seclabel,relatime 0 0
   selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
   debugfs /sys/kernel/debug debugfs rw,relatime 0 0
   none /var tmpfs rw,seclabel,relatime,mode=770,gid=1000
   0 0
   none /acct cgroup rw,relatime,cpuacct 0 0
   none /sys/fs/cgroup tmpfs
   rw,seclabel,relatime,mode=750,gid=1000 0 0
   none /sys/fs/cgroup/memory cgroup rw,relatime,memory 0
   0
   tmpfs /mnt/asec tmpfs
   rw,seclabel,relatime,mode=755,gid=1000 0 0
   tmpfs /mnt/obb tmpfs
   rw,seclabel,relatime,mode=755,gid=1000 0 0
   none /dev/memcg cgroup rw,relatime,memory 0 0
   none /dev/cpuctl cgroup rw,relatime,cpu 0 0
   ```

```
none /sys/fs/cgroup tmpfs
rw,seclabel,relatime,mode=750,gid=1000 0 0
none /sys/fs/cgroup/memory cgroup rw,relatime,memory 0
0
none /sys/fs/cgroup/freezer cgroup rw,relatime,freezer
0 0
/dev/block/platform/msm_sdcc.1/by-name/system /system
ext4 ro,seclabel,relatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/userdata /data
ext4
rw,seclabel,nosuid,nodev,relatime,noauto_da_alloc,data
=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/cache /cache
ext4 rw,seclabel,nosuid,nodev,relatime,data=ordered 0
0
/dev/block/platform/msm_sdcc.1/by-name/persist
/persist ext4
rw,seclabel,nosuid,nodev,relatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/modem /firmware
vfat
ro,context=u:object_r:firmware_file:s0,relatime,uid=10
00,gid=1000,fmask=0337,dmask=0227,codepage=cp437,iocha
rset=iso8859-1,shortname=lower,errors=remount-ro 0 0
/dev/fuse /mnt/shell/emulated fuse
rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,de
fault_permissions,allow_other 0 0
/dev/fuse /mnt/shell/emulated/0 fuse
rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,de
fault_permissions,allow_other 0 0
```

找到其中我们关注的带 /system 的那一行：

```
/dev/block/platform/msm_sdcc.1/by-name/system /system
ext4 ro,seclabel,relatime,data=ordered 0 0
```

3. 重新挂载。

命令：

```
mount -o remount,rw -t yaffs2
/dev/block/platform/msm_sdcc.1/by-name/system /system
```

这里的 `/dev/block/platform/msm_sdcc.1/by-name/system` 就是我们从上一步的输出里得到的文件路径。

如果输出没有提示错误的话，操作就成功了，可以对 /system 下的文件为所欲为了。

查看连接过的 WiFi 密码

注：需要 root 权限。

命令：

```
adb shell
su
cat /data/misc/wifi/*.conf
```

输出示例：

```
network={
    ssid="TP-LINK_9DFC"
    scan_ssid=1
    psk="123456789"
    key_mgmt=WPA-PSK
    group=CCMP TKIP
    auth_alg=OPEN
    sim_num=1
    priority=13893
}

network={
    ssid="TP-LINK_F11E"
    psk="987654321"
```

```
    key_mgmt=WPA-PSK
    sim_num=1
    priority=17293
}
```

`ssid` 即为我们在 WLAN 设置里看到的名称，`psk` 为密码，`key_mgmt` 为安全加密方式。

设置系统日期和时间

注：需要 root 权限。

命令：

```
adb shell
su
date -s 20160823.131500
```

表示将系统日期和时间更改为 2016 年 08 月 23 日 13 点 15 分 00 秒。

重启手机

命令：

```
adb reboot
```

检测设备是否已 root

命令：

```
adb shell
su
```

此时命令行提示符是 `$` 则表示没有 root 权限，是 `#` 则表示已 root。

使用 Monkey 进行压力测试

Monkey 可以生成伪随机用户事件来模拟单击、触摸、手势等操作，可以对正在开发中的程序进行随机压力测试。

简单用法：

```
adb shell monkey -p <packagename> -v 500
```

表示向 `<packagename>` 指定的应用程序发送 500 个伪随机事件。

Monkey 的详细用法参考 官方文档。

10. 刷机相关命令

重启到 Recovery 模式

命令：

```
adb reboot recovery
```

从 Recovery 重启到 Android

命令：

```
adb reboot
```

重启到 Fastboot 模式

命令：

```
adb reboot bootloader
```

通过 sideload 更新系统

如果我们下载了 Android 设备对应的系统更新包到电脑上，那么也可以通过 adb 来完成更新。

以 Recovery 模式下更新为例：

1. 重启到 Recovery 模式。

   命令：

   ```
   adb reboot recovery
   ```

2. 在设备的 Recovery 界面上操作进入 `Apply update-Apply from ADB`。

   注：不同的 Recovery 菜单可能与此有差异，有的是一级菜单就有 `Apply update from ADB`。

3. 通过 adb 上传和更新系统。

   命令：

   ```
   adb sideload <path-to-update.zip>
   ```

11. 更多 adb shell 命令

Android 系统是基于 Linux 内核的，所以 Linux 里的很多命令在 Android 里也有相同或类似的实现，在 `adb shell` 里可以调用。本文档前面的部分内容已经用到了 `adb shell` 命令。

查看进程

命令：

```
adb shell ps
```

输出示例：

```
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     8904   788   ffffffff 00000000 S /init
root      2     0     0      0     ffffffff 00000000 S kthreadd
...
u0_a71    7779  5926  1538748 48896 ffffffff 00000000 S
com.sohu.inputmethod.sogou:classic
u0_a58    7963  5926  1561916 59568 ffffffff 00000000 S
org.mazhuang.boottimemeasure
...
shell     8750  217   10640  740   00000000 b6f28340 R ps
```

各列含义：

列名	含义
USER	所属用户

列名	含义
PID	进程 ID
PPID	父进程 ID
NAME	进程名

查看实时资源占用情况

命令：

```
adb shell top
```

输出示例：

```
User 0%, System 6%, IOW 0%, IRQ 0%
User 3 + Nice 0 + Sys 21 + Idle 280 + IOW 0 + IRQ 0 + SIRQ
3 = 307

  PID PR CPU% S  #THR    VSS    RSS PCY UID      Name
 8763  0   3% R    1  10640K   1064K  fg shell    top
  131  0   3% S    1      0K      0K  fg root     dhd_dpc
 6144  0   0% S  115 1682004K 115916K  fg system
system_server
  132  0   0% S    1      0K      0K  fg root     dhd_rxf
 1731  0   0% S    6  20288K    788K  fg root
/system/bin/mpdecision
  217  0   0% S    6  18008K    356K  fg shell    /sbin/adbd
 ...
 7779  2   0% S   19 1538748K  48896K  bg u0_a71
com.sohu.inputmethod.sogou:classic
 7963  0   0% S   18 1561916K  59568K  fg u0_a58
org.mazhuang.boottimemeasure
 ...
```

各列含义：

列名	含义
PID	进程 ID
PR	优先级
CPU%	当前瞬间占用 CPU 百分比
S	进程状态（R=运行，S=睡眠，T=跟踪/停止，Z=僵尸进程）
#THR	线程数
VSS	Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）
RSS	Resident Set Size 实际使用物理内存（包含共享库占用的内存）
PCY	调度策略优先级，SP_BACKGROUND/SPFOREGROUND
UID	进程所有者的用户 ID
NAME	进程名

`top` 命令还支持一些命令行参数，详细用法如下：

```
Usage: top [ -m max_procs ] [ -n iterations ] [ -d delay ] [ -s sort_column ] [ -t ] [ -h ]
   -m num  最多显示多少个进程
   -n num  刷新多少次后退出
   -d num  刷新时间间隔（单位秒，默认值 5）
   -s col  按某列排序（可用 col 值：cpu, vss, rss, thr）
```

-t	显示线程信息
-h	显示帮助文档

其它

如下是其它常用命令的简单描述，前文已经专门讲过的命令不再额外说明：

命令	功能
cat	显示文件内容
cd	切换目录
chmod	改变文件的存取模式/访问权限
df	查看磁盘空间使用情况
grep	过滤输出
kill	杀死指定 PID 的进程
ls	列举目录内容
mount	挂载目录的查看和管理
mv	移动或重命名文件
ps	查看正在运行的进程
rm	删除文件

命令	功能
top	查看进程的资源占用情况