ADB 操作命令详解及用法大全

一、ADB 是什么?

ADB，即 Android Debug Bridge 是一种允许模拟器或已连接的 Android 设备进行通信的命令行工具，它可为各种设备操作提供便利，如安装和调试应用，并提供对 `Unix shell` （可用来在模拟器或连接的设备上运行各种命令）的访问。可以在 `Android SDK/platform-tools` 中找到 `adb` 工具或下载 ADB Kits 。

注： 有部分命令的支持情况可能与 Android 系统版本及定制 ROM 的实现有关。

二、ADB 有什么作用?

`ADB` 是 `Android SDK` 里的一个工具, 用这个工具可以直接操作管理 Android 模拟器或者真实的 Android 设备。它的主要功能有:

- 在设备上运行 Shell 命令；
- 将本地 APK 软件安装至模拟器或 Android 设备；
- 管理设备或手机模拟器上的预定端口；
- 在设备或手机模拟器上复制或粘贴文件。

`ADB` 是一个 `客户端-服务器程序` 程序，包括三个组件：

- 客户端：该组件发送命令。客户端在开发计算机上运行。您可以通过发出 adb 命令从命令行终端调用客户端。

- 后台程序：该组件在设备上运行命令。后台程序在每个模拟器或设备实例上作为后台进程运行。
- 服务器：该组件管理客户端和后台程序之间的通信。服务器在开发计算机上作为后台进程运行。

三、ADB 命令语法

adb 命令的基本语法如下：

```
adb [-d|-e|-s <serial-number>] <command>
复制代码
```

单一设备 / 模拟器连接

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

多个设备 / 模拟器连接

如果有多个设备 / 模拟器连接，则需要为命令指定目标设备，下表是指定目标设备的命令选项：

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 <serial-number>	指定相应设备序列号的设备 / 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 <serial-number>` 参数， `serial-number` 是指设备的设备序列号，可以通过 `adb devices` 命令获取。

四、ADB 常用命令

4.1 基本命令

4.1.1 查看 adb 的版本信息

```
adb version
复制代码
```

4.1.2 启动 adb

```
adb start-server
复制代码
```

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。

4.1.3 停止 adb

```
adb kill-server
复制代码
```

4.1.4 以 root 权限运行 adbd

```
adb root
复制代码
```

4.1.5 指定 adb server 的网络端口

```
adb -P <port> start-server
复制代码
```

ADB 的默认端口为 5037。

4.1.5 查询已连接的设备 / 模拟器列表

```
adb devices
复制代码
```

4.2 设备连接管理

4.2.1 USB 连接

通过 USB 连接来正常使用 adb 需要以下步骤：

1. 确认硬件状态正常 (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 (可以在「设置」-「开发者选项」-「USB 调试」打开 USB 调试)。
3. 确认设备驱动状态正常 (安装 ADB 驱动程序)。
4. 通过 USB 线连接好电脑和设备后确认状态。
5. 通过 `adb devices` 命令查看设备连接情况。

4.2.2 WLAN 连接（需要 USB 线）

借助 USB 通过 WiFi 连接来正常使用 adb 需要以下步骤： 操作步骤：

1. 将 Android 设备与要运行 adb 的电脑连接到同一个 WiFi。
2. 将设备与电脑通过 USB 线连接 (可通过 `adb devices` 命令查看设备连接情况)。
3. 通过 `adb tcpip 5555` 命令让设备在 5555 端口监听 TCP/IP 连接。
4. 断开 USB 连接。
5. 找到设备的 IP 地址 (可以在「设置」-「关于手机」-「状态信息」-「IP 地址」查看 IP 地址)。
6. 通过 `adb connect <device-ip-address>` 命令使用 IP 地址将 Android 设备与电脑连接。
7. 通过 `adb devices` 命令查看设备连接情况。
8. 使用完毕后可通过 `adb disconnect <device-ip-address>` 命令断开无线连接。

4.2.3 WLAN 连接（无需借助 USB 线）

注：需要 root 权限。 不借助 USB 通过 WiFi 连接来正常使用 adb 需要以下步骤：

1. 在 Android 设备上安装一个终端模拟器 (可通过 Terminal Emulator for Android Downloads 下载)。
2. 将 Android 设备与要运行 adb 的电脑连接到同一个 WiFi。

3. 打开 Android 设备上的终端模拟器，在里面依次运行命令：

```
su
setprop service.adb.tcp.port 5555
复制代码
```

4. 找到设备的 IP 地址 (可以在「设置」-「关于手机」-「状态信息」-「IP 地址」查看 IP 地址)。

5. 通过 `adb connect <device-ip-address>` 命令使用 IP 地址将 Android 设备与电脑连接。

6. 通过 `adb devices` 命令查看设备连接情况。

4.2.4 WiFi 连接转为 USB 连接

通过 `adb usb` 命令以 USB 模式重新启动 ADB：

```
adb usb
复制代码
```

4.3 应用管理

4.3.1 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 U
复制代码
```

`adb shell pm list packages` 后面可以跟一些可选参数进行过滤查看不同的列表，可用参数及含义如下：

参数	显示列表
无	所有应用
-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

参数	显示列表
-i	显示应用的 installer
-u	包含已卸载应用
<filter>	包名包含 <filter> 字符串

4.3.1.1 查看所有应用

```
adb shell pm list packages
```
复制代码

4.3.1.2 查看系统应用

```
adb shell pm list packages -s
```
复制代码

4.3.1.3 查看第三方应用

```
adb shell pm list packages -3
```
复制代码

4.3.1.4 包名包含某字符串的应用

比如要查看包名包含字符串 `huawei` 的应用列表，命令：

```
adb shell pm list packages huawei
```
复制代码

4.3.2 安装应用

安装应用的基本命令格式是：

```
adb install [-l] [-r] [-t] [-s] [-d] [-g] <apk-file>
```
复制代码

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

参数	含义
-l	将应用安装到保护目录 /mnt/asec
-r	允许覆盖安装

参数	含义
-t	允许安装 AndroidManifest.xml 里 application 指定 `android:testOnly="true"` 的应用
-s	将应用安装到 sdcard
-d	允许降级覆盖安装
-g	授予所有运行时权限

运行命令后可以看到输出内容，包含安装进度和状态，安装状态如下：

- `Success` ：代表安装成功。
- `Failure` ：代表安装失败。 APK 安装失败的情况有很多，`Failure` 状态之后有安装失败输出代码。常见安装失败输出代码、含义及可能的解决办法如下：

输出代码	含义
INSTALL_FAILED_ALREADY_EXISTS	应用已经存在，或卸载了
INSTALL_FAILED_INVALID_APK	无效的 APK 文件
INSTALL_FAILED_INVALID_URI	无效的 APK 文件名
INSTALL_FAILED_INSUFFICIENT_STORAGE	空间不足
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	以前安装过同名应用， 但卸载时数据没有移除； 但签名不一致
INSTALL_FAILED_SHARED_USER_INCOMPATIBLE	请求的共享用户存在但签
INSTALL_FAILED_MISSING_SHARED_LIBRARY	安装包使用了设备上不可

输出代码	含义
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
INSTALL_FAILED_OLDER_SDK	设备系统版本低于应用要
INSTALL_FAILED_CONFLICTING_PROVIDER	设备里已经存在与应用里 provider
INSTALL_FAILED_NEWER_SDK	设备系统版本高于应用要
INSTALL_FAILED_TEST_ONLY	应用是 test-only 的，但 参数
INSTALL_FAILED_CPU_ABI_INCOMPATIBLE	包含不兼容设备 CPU 应 native code
INSTALL_FAILED_MISSING_FEATURE	应用使用了设备不可用的
INSTALL_FAILED_CONTAINER_ERROR	1. sdcard 访问失败; 2. 应用签名与 ROM 签名 被当作内置应用。
INSTALL_FAILED_INVALID_INSTALL_LOCATION	1. 不能安装到指定位置; 2. 应用签名与 ROM 签名 被当作内置应用。
INSTALL_FAILED_MEDIA_UNAVAILABLE	安装位置不可用
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
INSTALL_FAILED_UID_CHANGED	以前安装过该应用，与本 不一致
INSTALL_FAILED_VERSION_DOWNGRADE	已经安装了该应用更高版

输出代码	含义
INSTALL_FAILED_PERMISSION_MODEL_DOWNGRADE	已安装 target SDK 支持运行时权限的同名应 要安装的版本不支持运行
INSTALL_PARSE_FAILED_NOT_APK	指定路径不是文件，或不
INSTALL_PARSE_FAILED_BAD_MANIFEST	无法解析的 AndroidMan
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
INSTALL_PARSE_FAILED_CERTIFICATE_ENCODING	解析 APK 文件时遇到 CertificateEncodin
INSTALL_PARSE_FAILED_BAD_PACKAGE_NAME	manifest 文件里没有或者
INSTALL_PARSE_FAILED_BAD_SHARED_USER_ID	manifest 文件里指定了无
INSTALL_PARSE_FAILED_MANIFEST_MALFORMED	解析 manifest 文件时遇到
INSTALL_PARSE_FAILED_MANIFEST_EMPTY	在 manifest 文件里找不到 （instrumentation 或 ap
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
INSTALL_FAILED_NO_MATCHING_ABIS	应用包含设备的应用程序 native code
INSTALL_CANCELED_BY_USER	应用安装需要在设备上确 但未操作设备或点了取消
INSTALL_FAILED_ACWF_INCOMPATIBLE	应用程序与设备不兼容
INSTALL_FAILED_TEST_ONLY	APK 文件是使用 Android 编译出来的文件

输出代码	含义
does not contain AndroidManifest.xml	无效的 APK 文件
is not a valid zip file	无效的 APK 文件
Offline	设备未连接成功
unauthorized	设备未授权允许调试
error: device not found	没有连接成功的设备
protocol failure	设备已断开连接
Unknown option: -s	Android 2.2 以下不支持
No space left on device	空间不足
Permission denied ... sdcard ...	sdcard 不可用
signatures do not match the previously installed version; ignoring!	已安装该应用且签名不一

参考： PackageManager.java

`adb install` 实际是分三步完成：

1. push apk 文件到 /data/local/tmp。
2. 调用 pm install 安装。
3. 删除 /data/local/tmp 下的对应 apk 文件。

4.3.3 卸载应用

卸载应用的基本命令格式是：

```
adb uninstall [-k] <package-name>
复制代码
```

`<package-name>` 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。

4.3.4 清除应用数据与缓存

```
adb shell pm clear <package-name>
```
复制代码

`<package-name>` 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」。

4.3.5 查看前台 Activity

```
adb shell dumpsys activity activities | grep mFocusedActivity
```
复制代码

4.3.6 查看正在运行的 Services

```
adb shell dumpsys activity services [<package-name>]
```
复制代码

`<package-name>` 参数不是必须的，指定 `<package-name>` 表示查看与某个包名相关的 Services，不指定表示查看所有 Services。`<package-name>` 不一定要给出完整的包名，可以只给一部分，那么所给包名相关的 Services 都会列出来。

4.3.7 查看应用详细信息

```
adb shell dumpsys package <package-name>
```
复制代码

`<package-name>` 表示应用包名。运行次命令的输出中包含很多信息，包括 Activity Resolver Table、Registered ContentProviders、包名、userId、安装后的文件资源代码等路径、版本信息、权限信息和授予状态、签名版本信息等。

4.3.7 查看应用安装路径

```
adb shell pm path <package-name>
```
复制代码

4.4 与应用交互

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 <package-name>`	停止 <package-name> 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

`<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

参数	含义
`--ei <extra-key> <extra-int-value>`	integer 值
`--el <extra-key> <extra-long-value>`	long 值
`--ef <extra-key> <extra-float-value>`	float 值
`--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

4.4.1 启动应用 / 调起 Activity

```
adb shell am start [options] <intent>
```
复制代码

例如：

```
adb shell am start -a android.settings.SETTINGS                 # 打开
adb shell am start -a android.intent.action.DIAL -d tel:10086   # 打开
adb shell am start -n com.android.mms/.ui.ConversationList      # 打开
```
复制代码

options 是一些改变其行为的选项，支持的可选参数及含义如下：

选项	含义
-D	启用调试
-W	等待启动完成
--start-profiler file	启动分析器并将结果发送到 file

选项	含义
-P `file`	类似于 --start-profiler， 但当应用进入空闲状态时分析停止
-R `count`	重复 Activity 启动次数
-S	启动 Activity 前强行停止目标应用
--opengl-trace	启用 OpenGL 函数的跟踪
--user `user_id` | current	指定要作为哪个用户运行；如果未指定， 则作为当前用户运行

4.4.2 调起 Service

```
adb shell am startservice [options] <intent>
```
复制代码

一个典型的用例是如果设备上原本应该显示虚拟按键但是没有显示，可以试试这个：

```
adb shell am startservice -n com.android.systemui/.SystemUIService
```
复制代码

4.4.3 停止 Service

```
adb shell am stopservice [options] <intent>
```
复制代码

4.4.4 发送广播

```
adb shell am broadcast [options] <INTENT>
```
复制代码

可以向所有组件广播，也可以只向指定组件广播。

例如，向所有组件广播 `BOOT_COMPLETED` ：

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED
```
复制代码

又例如，只向 `com.android.receiver.test/.BootCompletedReceiver` 广播 `BOOT_COMPLETED` ：

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -n com.a
```
复制代码

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。

既能发送系统预定义的广播，也能发送自定义广播。如下是部分系统预定义广播及正常触发时机：

action	触发时机
android.net.conn.CONNECTIVITY_CHANGE	网络连接发生变化
android.intent.action.SCREEN_ON	屏幕点亮
android.intent.action.SCREEN_OFF	屏幕熄灭
android.intent.action.BATTERY_LOW	电量低， 会弹出电量低提示框
android.intent.action.BATTERY_OKAY	电量恢复了
android.intent.action.BOOT_COMPLETED	设备启动完毕
android.intent.action.DEVICE_STORAGE_LOW	存储空间过低
android.intent.action.DEVICE_STORAGE_OK	存储空间恢复
android.intent.action.PACKAGE_ADDED	安装了新的应用
android.net.wifi.STATE_CHANGE	WiFi 连接状态发生变化
android.net.wifi.WIFI_STATE_CHANGED	WiFi 状态变为启用 / 关闭 / 正在启动 / 正在关闭 / 未知
android.intent.action.BATTERY_CHANGED	电池电量发生变化
android.intent.action.INPUT_METHOD_CHANGED	系统输入法发生变化
android.intent.action.ACTION_POWER_CONNECTED	外部电源连接
android.intent.action.ACTION_POWER_DISCONNECTED	外部电源断开连接
android.intent.action.DREAMING_STARTED	系统开始休眠

action	触发时机
android.intent.action.DREAMING_STOPPED	系统停止休眠
android.intent.action.WALLPAPER_CHANGED	壁纸发生变化
android.intent.action.HEADSET_PLUG	插入耳机
android.intent.action.MEDIA_UNMOUNTED	卸载外部介质
android.intent.action.MEDIA_MOUNTED	挂载外部介质
android.os.action.POWER_SAVE_MODE_CHANGED	省电模式开启

（以上广播均可使用 adb 触发）

4.4.5 强制停止应用

```
adb shell am force-stop <packagename>
```
复制代码

4.4.6 收紧内存

```
adb shell am send-trim-memory  <pid> <level>
```
复制代码

参数说明：

- `pid` ：进程 ID
- `level` :HIDDEN、RUNNING_MODERATE、BACKGROUND、RUNNING_LOW、MODERATE、RUNNING_CRITICAL、COMPLETE

4.5 文件管理

4.5.1 从模拟器 / 设备下载指定的文件到计算机

从模拟器 / 设备下载指定的文件到计算机的基本命令格式是：

```
adb pull <remote> [local]
```
复制代码

参数说明：

- `remote` ：模拟器 / 设备里的文件路径

- `local`：计算机上的目录，参数可以省略，默认复制到当前目录

例如，将 /sdcard/music.mp4 下载到计算机的当前目录：

```
adb pull /sdcard/music.mp4
```
复制代码

将 /sdcard/music.mp4 下载到计算机的当前目录 (目录需存在)：

```
adb pull /sdcard/music.mp4 D:\Download
```
复制代码

4.5.2 将指定的文件从计算机上传到模拟器 / 设备

将指定的文件从计算机上传到模拟器 / 设备的基本命令格式是：

```
adb push <local> <remote>
```
复制代码

参数说明：

- `local`：计算机上的文件路径
- `remote`：模拟器 / 设备里的目录

例如，将 D:\Download\music.mp4 下载到设备的 / sdcard/music / 目录：

```
adb push D:\Download\music.mp4 /sdcard/music/
```
复制代码

4.5.4 列出指定目录的内容

列出模拟器 / 设备上指定目录的内容的基本命令格式是：

```
adb shell ls [options] <directory>
```
复制代码

`<directory>` 表示指定目录，可以省略，表示列出根目录下的所有文件和目录。 `adb shell ls` 后面可以跟一些可选参数进行过滤查看不同的列表，可用参数及含义如下：

参数	显示列表
无	列出目录下的所有文件和目录

参数	显示列表
-a	列出目录下的所有文件 (包括隐藏的)
-i	列出目录下的所有文件和索引编号
-s	列出目录下的所有文件和文件大小
-n	列出目录下的所有文件及其 UID 和 GID
-R	列出目录下的所有子目录中的文件

4.5.5 切换到目标目录

```
adb shell cd <directory>
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `cd <directory>` 命令切换到目标目录。

4.5.6 删除文件或目录

```
adb shell rm [options] <files or directory>
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `rm [options] <files or directory>` 命令删除文件或目录。

`rm` 后面可以跟一些可选参数进行不同的操作，可用参数及含义如下：

参数	含义
无	删除文件
-f	强制删除文件，系统不提示
-r	强制删除指定目录中的所有文件和子目录
-d	删除指定目录，即使它是一个非空目录
-i	交互式删除，删除前提示

`rm -d` 等同于 `rmdir` 命令，有些版本不包含 `-d` 参数。

4.5.7 创建目录

```
adb shell mkdir [options] <directory-name>
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `mkdir [options] <directory-name>` 命令创建目录。 `mkdir` 后面可以跟一些可选参数进行不同的操作，可用参数及含义如下：

参数	含义
无	创建指定目录
-m	创建指定目录并赋予读写权限
-p	创建指定目录及其父目录

4.5.8 创建空文件或改变文件时间戳

```
adb shell touch [options] <file>
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `touch [options] <file>` 命令创建空文件或改变文件时间戳。

可通过 `ls -n <directory>` 命令查看文件的时间。

4.5.9 输出当前目录路径

```
adb shell pwd
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `pwd` 命令输出当前目录路径。

4.5.10 复制文件和目录

```
adb shell cp [options] <source> <dest>
```
复制代码

第一步：执行 `adb shell` 命令； 第二步：执行 `cp [options] <source> <dest>` 命令复制文件和目录。 参数说明：

- `source` ：源文件路径
- `dest` ：目标文件路径

4.5.11 移动或重命名文件

```
adb shell mv [options] <source> <dest>
复制代码
```

第一步：执行 `adb shell` 命令； 第二步：执行 `mv [options] <source> <dest>` 命令移动或重命名文件。 参数说明：

- `source` ：源文件路径
- `dest` ：目标文件路径

4.6 网络管理

4.6.1 查看网络统计信息

```
adb shell netstat
复制代码
```

也可以将网络统计信息输出到指定文件：

```
adb shell netstat><file-path>
复制代码
```

例如，可以通过 `adb shell netstat>D:\netstat.log` 将日志输出到 `D:\netstat.log` 中。

4.6.2 测试两个网络间的连接和延迟

`ping` 命令的格式如下：

```
adb shell ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interf
[-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
[-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
[-w deadline] [-W timeout] [hop1 ...] destination
复制代码
```

例如，ping 一个域名：

```
adb shell ping www.google.com
复制代码
```

不结束的话会一直 ping 下去，可以按 `Ctrl + C` 停止 ping 操作。

也可以指定 ping 的次数：

```
adb shell ping -c 4 www.google.com
```
复制代码

4.6.3 通过配置文件配置和管理网络连接

`netcfg` 命令的格式如下：

```
adb shell netcfg [<interface> {dhcp|up|down}]
```
复制代码

输出示例：

```
rmnet_ims10 DOWN                                  0.0.0.0/0   0x000010(
rmnet_ims00 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun04 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun03 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun02 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun01 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun00 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun14 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun13 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun12 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun11 DOWN                                  0.0.0.0/0   0x000010(
rmnet_tun10 DOWN                                  0.0.0.0/0   0x000010(
rmnet1    DOWN                                 0.0.0.0/0   0x00001002
rmnet0    DOWN                                 0.0.0.0/0   0x00001002
rmnet4    DOWN                                 0.0.0.0/0   0x00001002
rmnet3    DOWN                                 0.0.0.0/0   0x00001002
rmnet2    DOWN                                 0.0.0.0/0   0x00001002
rmnet6    DOWN                                 0.0.0.0/0   0x00001002
rmnet5    DOWN                                 0.0.0.0/0   0x00001002
dummy0    UP                                   0.0.0.0/0   0x000000c3
rmnet_r_ims10 DOWN                                  0.0.0.0/0   0x0000
rmnet_r_ims00 DOWN                                  0.0.0.0/0   0x0000
rmnet_emc0 DOWN                                  0.0.0.0/0   0x0000100
lo        UP                                 127.0.0.1/8   0x00000049
sit0      DOWN                                 0.0.0.0/0   0x00000080
wlan0     UP                                10.0.38.176/23  0x00001043
```
复制代码

4.6.4 显示、操作路由、设备、策略路由和隧道

`ip` 命令的格式如下：

```
adb shell ip [ options ] object
```
复制代码

- options := {-V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |-f[amily] { inet | inet6 | ipx | dnet | link } |-l[oops] {

maximum-addr-flush-attempts } |-o[neline] | -t[imestamp] | -b[atch] [filename] |-rc[vbuf] [size]}

- object := {link | addr | addrlabel | route | rule | neigh | ntable |tunnel | tuntap | maddr | mroute | mrule | monitor | xfrm |netns | l2tp}

`options` 是一些修改 ip 行为或者改变其输出的选项。所有的选项都是以 - 字符开头，分为长、短两种形式，支持的可选参数及含义如下：

选项	含义
-V,-Version	打印 ip 的版本并退出
-s,-stats,-statistics	输出更为详尽的信息 (如果这个选项出现两次或者多次， 输出的信息将更为详尽)
-f,-family	强调使用的协议种类 (包括：inet、inet6 或者 link)
-4	是 - family inet 的简写
-6	是 - family inet6 的简写
-0	是 - family link 的简写
-o,-oneline	对每行记录都使用单行输出，回行用字符代替
-r,-resolve	查询域名解析系统，用获得的主机名代替主机 IP 地址

`object` 是你要管理或者获取信息的对象。目前 ip 认识的对象包括：

参数	显示列表
link	网络设备
address	一个设备的协议 (IP 或者 IPV6) 地址
neighbour	ARP 或者 NDISC 缓冲区条目

参数	显示列表
route	路由表条目
rule	路由策略数据库中的规则
maddress	多播地址
mroute	多播路由缓冲区条目
tuntap	管理 TUN/TAP 设备
netns	管理网络空间

例如，查看 `WiFi IP` 地址：

```
adb shell ip -f inet addr show wlan0
```
复制代码

4.7 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。 可以执行 `adb shell input` 命令查看完整 help 信息如下：

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

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

The commands and default sources are:
      text <string> (Default: touchscreen)
      keyevent [--longpress] <key code number or name> ... (Default: ke
      tap <x> <y> (Default: touchscreen)
      swipe <x1> <y1> <x2> <y2> [duration(ms)] (Default: touchscreen)
      draganddrop <x1> <y1> <x2> <y2> [duration(ms)] (Default: touchscr
      press (Default: trackball)
      roll <dx> <dy> (Default: trackball)
```
复制代码

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
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

keycode	含义
207	打开联系人
208	打开日历
209	打开音乐
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。

4.7.1 电源键

```
adb shell input keyevent 26
```
复制代码

执行效果相当于按电源键。

4.7.2 菜单键

```
adb shell input keyevent 82
```
复制代码

4.7.3 HOME 键

```
adb shell input keyevent 3
```
复制代码

4.7.4 返回键

```
adb shell input keyevent 4
```
复制代码

4.7.5 音量控制

- 增加音量：

```
adb shell input keyevent 24
```
复制代码

- 降低音量：

```
adb shell input keyevent 25
```
复制代码

- 静音：

```
adb shell input keyevent 164
```
复制代码

4.7.6 媒体控制

- 播放 / 暂停：

```
adb shell input keyevent 85
```
复制代码

- 停止播放：

```
adb shell input keyevent 86
```
复制代码

- 播放下一首：

```
adb shell input keyevent 87
```
复制代码

- 播放上一首：

```
adb shell input keyevent 88
```
复制代码

- 恢复播放：

```
adb shell input keyevent 126
```
复制代码

- 暂停播放：

```
adb shell input keyevent 127
```
复制代码

4.7.7 点亮 / 熄灭屏幕

- 点亮屏幕：

```
adb shell input keyevent 224
```
复制代码

- 熄灭屏幕：

```
adb shell input keyevent 223
```
复制代码

4.7.8 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
```
复制代码

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` 。

4.7.9 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。

```
adb shell input text hello
```
复制代码

4.8 日志打印

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。

4.8.1 Android 日志

查看 Android 设备系统属性的基本命令格式是：

```
adb logcat [option] [filter-specs]
```
复制代码

如果需要停止 `logcat` 日志打印，可以按 `Ctrl + C` 停止日志监控。

4.8.1.1 按级别过滤日志

按级别过滤日志的基本命令格式是：

```
adb logcat [filter-specs]
```
复制代码

Android 的日志分为如下几个优先级（priority）：

级别	含义
*:V	过滤只显示 Verbose 及以上级别 (优先级最低)
*:D	过滤只显示 Debug 及以上级别
*:I	过滤只显示 Info 及以上级别
*:W	过滤只显示 Warning 及以上级别
*:E	过滤只显示 Error 及以上级别
*:F	过滤只显示 Fatal 及以上级别
*:S	过滤只显示 Silent 及以上级别 (优先级最高，什么也不输出)

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
```
复制代码

会将 Warning、Error、Fatal 和 Silent 日志输出。

（注：在 macOS 下需要给 `*:W` 这样以 `*` 作为 tag 的参数加双引号，如 `adb logcat "*:W"`，不然会报错 `no matches found: *:W`。）

4.8.1.2 按 tag 和级别过滤日志

按 tag 和级别过滤日志的基本命令格式是：

```
adb logcat [tag:level] [tag:level] ...
```
复制代码

比如，命令：

```
adb logcat ActivityManager:I MyApp:D *:S
```
复制代码

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）。

4.8.1.3 将日志格式化输出

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>` ：

参数	显示格式
brief	`<priority>/<tag>(<pid>): <message>`
process	`<priority>(<pid>) <message>`
tag	`<priority>/<tag>: <message>`
raw	`<message>`
time	`<datetime> <priority>/<tag> (<pid>): <message>`
threadtime	`<datetime> <pid> <tid> <priority> <tag>: <message>`
long	`[ <datetime> <pid>:<tid> <priority>/<tag> ] <message>`

日志格式默认为 `brief` ，指定格式可与上面的过滤同时使用。比如：

```
adb logcat -v long ActivityManager:I *:S
```
复制代码

4.8.1.3 清空已存在的日志

```
adb logcat -c
```
复制代码

4.8.1.4 将日志显示在控制台

```
adb logcat -d
```
复制代码

4.8.1.5 将日志输出到文件

```
adb logcat -f <file-path>
```
复制代码

4.8.1.6 加载一个可使用的日志缓冲区供查看

```
adb logcat -b <Buffer>
```
复制代码

`Android log` 输出量巨大，特别是通信系统的 log，因此，Android 把 log 输出到不同的缓冲区中，目前定义了四个 log 缓冲区：

缓冲区	含义
Radio	输出通信系统的 log
System	输出系统组件的 log
Event	输出 event 模块的 log
Main	所有 java 层的 log 以及不属于上面 3 层的 log

缓冲区主要给系统组件使用，一般的应用不需要关心，应用的 log 都输出到 main 缓冲区中。默认 log 输出（不指定缓冲区的情况下）是输出 System 和 Main 缓冲区的 log。

4.8.1.7 打印指定日志缓冲区的大小

```
adb logcat -g
```
复制代码

4.8.2 内核日志

```
adb shell dmesg
```
复制代码

输出示例：

```
<6>[14201.872498] PM: Syncing filesystems ... done.
```
复制代码

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。

4.9 查看 Android 设备系统属性

查看 Android 设备系统属性的基本命令格式是：

```
adb shell getprop [options]
```
复制代码

除了可以查看 Android 设备系统属性之外，还可以设置系统属性，设置系统属性的基本命令格式是：

```
adb shell setprop <key> <value>
```
复制代码

4.9.1 查看设备型号

```
adb shell getprop ro.product.model
```
复制代码

输出示例：

```
Nexus 5
```
复制代码

4.9.2 查看设备电池状况

```
adb shell dumpsys battery
```
复制代码

输出示例：

```
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
复制代码
```

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。

4.9.3 查看设备屏幕分辨率

```
adb shell wm size
复制代码
```

输出示例：

```
Physical size: 1080x1920
复制代码
```

该设备屏幕分辨率为 1080px * 1920px。

如果使用命令修改过，那输出可能是：

```
Physical size: 1080x1920
Override size: 480x1024
复制代码
```

表明设备的屏幕分辨率原本是 1080px * 1920px，当前被修改为 480px * 1024px。

4.9.4 查看设备屏幕密度

```
adb shell wm density
复制代码
```

输出示例：

```
Physical density: 420
复制代码
```

该设备屏幕密度为 420dpi。

如果使用命令修改过，那输出可能是：

```
Physical density: 480
Override density: 160
复制代码
```

表明设备的屏幕密度原来是 480dpi，当前被修改为 160dpi。

4.9.5 查看设备显示屏参数

```
adb shell dumpsys window displays
```
复制代码

输出示例：

```
WINDOW MANAGER DISPLAY CONTENTS (dumpsys window displays)
  Display: mDisplayId=0
    init=1080x1920 420dpi cur=1080x1920 app=1080x1794 rng=1080x1017-181(
    deferred=false layoutNeeded=false
```
复制代码

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。

4.9.6 查看设备 android_id

```
adb shell settings get secure android_id
```
复制代码

输出示例：

```
51b6be48bac8c569
```
复制代码

4.9.7 查看设备 IMEI

在 Android 4.4 及以下版本可通过如下命令获取 IMEI：

```
adb shell dumpsys iphonesubinfo
```
复制代码

输出示例：

```
Phone Subscriber Info:
  Phone Type = GSM
  Device ID = 860955027785041
```
复制代码

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
```
复制代码

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

4.9.8 查看设备 Android 系统版本

```
adb shell getprop ro.build.version.release
```
复制代码

输出示例：

```
5.0.2
```
复制代码

4.9.9 查看设备 IP 地址

```
adb shell ifconfig | grep Mask
```
复制代码

在有的设备上这个命令没有输出，如果设备连着 WiFi，可以使用如下命令来查看局域网 IP：

```
adb shell ifconfig wlan0
```
复制代码

如果以上命令仍然不能得到期望的信息，那可以试试以下命令（部分系统版本里可用）：

```
adb shell netcfg
```
复制代码

4.9.10 查看设备 Mac 地址

```
adb shell cat /sys/class/net/wlan0/address
```
复制代码

输出示例：

```
f8:a9:d0:17:42:4d
```
复制代码

这查看的是局域网 Mac 地址，移动网络或其它连接的信息可以通过前面的小节「IP 地址」里提到的 `adb shell netcfg` 命令来查看。

4.9.11 查看设备 CPU 信息

```
adb shell cat /proc/cpuinfo
```
复制代码

4.9.12 查看设备内存信息

```
adb shell cat /proc/meminfo
```
复制代码

4.9.13 查看设备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
```
复制代码

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 [*节注一*]
persist.sys.isUsbOtgEnabled	是否支持 OTG

属性名	含义
dalvik.vm.heapsize	每个应用程序的内存上限
ro.sf.lcd_density	屏幕密度

节注一：

一些小厂定制的 ROM 可能修改过 CPU 支持的 abi 列表的属性名，如果用 `ro.product.cpu.abilist` 属性名查找不到，可以这样试试：

```
adb shell cat /system/build.prop | grep ro.product.cpu.abi
复制代码
```

示例输出：

```
ro.product.cpu.abi=armeabi-v7a
ro.product.cpu.abi2=armeabi
复制代码
```

4.10 修改设置

注： 修改设置之后，运行恢复命令有可能显示仍然不太正常，可以运行 `adb reboot` 重启设备，或手动重启。

修改设置的原理主要是通过 settings 命令修改 /data/data/com.android.providers.settings/databases/settings.db 里存放的设置值。

4.10.1 修改分辨率

```
adb shell wm size 480x1024
复制代码
```

表示将分辨率修改为 480px * 1024px。

恢复原分辨率命令：

```
adb shell wm size reset
复制代码
```

4.10.2 修改屏幕密度

```
adb shell wm density 160
复制代码
```

表示将屏幕密度修改为 160dpi。

恢复原屏幕密度命令：

```
adb shell wm density reset
复制代码
```

4.10.3 修改显示区域

```
adb shell wm overscan 0,0,0,200
复制代码
```

四个数字分别表示距离左、上、右、下边缘的留白像素，以上命令表示将屏幕底部 200px 留白。

恢复原显示区域命令：

```
adb shell wm overscan reset
复制代码
```

4.10.4 修改关闭 USB 调试模式

```
adb shell settings put global adb_enabled 0
复制代码
```

用命令恢复不了了，毕竟关闭了 USB 调试 adb 就连接不上 Android 设备了。 去设备上手动恢复吧：「设置」-「开发者选项」-「Android 调试」。

4.10.5 修改允许 / 禁止访问非 SDK API

允许访问非 SDK API：

```
adb shell settings put global hidden_api_policy_pre_p_apps 1
adb shell settings put global hidden_api_policy_p_apps 1
复制代码
```

禁止访问非 SDK API：

```
adb shell settings delete global hidden_api_policy_pre_p_apps
adb shell settings delete global hidden_api_policy_p_apps
复制代码
```

不需要设备获得 Root 权限。

命令最后的数字的含义：

值	含义
0	禁止检测非 SDK 接口的调用。该情况下， 日志记录功能被禁用，并且令 strict mode API，即 detectNonSdkApiUsage() 无效。不推荐。
1	仅警告——允许访问所有非 SDK 接口， 但保留日志中的警告信息，可继续使用 strick mode API。
2	禁止调用深灰名单和黑名单中的接口。
3	禁止调用黑名单中的接口， 但允许调用深灰名单中的接口。

4.10.6 修改状态栏和导航栏的显示隐藏

```
adb shell settings put global policy_control <key-values>
```
复制代码

`<key-values>` 可由如下几种键及其对应的值组成，格式为 `<key1>=<value1>:<key2>=<value2>` 。

key	含义
immersive.full	同时隐藏
immersive.status	隐藏状态栏
immersive.navigation	隐藏导航栏
immersive.preconfirms	?

这些键对应的值可则如下值用逗号组合：

value	含义
apps	所有应用
*	所有界面
package-name	指定应用
-package-name	排除指定应用

例如：

```
adb shell settings put global policy_control immersive.full=*
复制代码
```

表示设置在所有界面下都同时隐藏状态栏和导航栏。

```
adb shell settings put global policy_control immersive.status=com.packa
复制代码
```

表示设置在包名为 `com.package1` 和 `com.package2` 的应用里隐藏状态栏，在除了包名为 `com.package3` 的所有应用里隐藏导航栏。

4.11 实用功能

4.11.1 屏幕截图

截图保存到电脑：

```
adb exec-out screencap -p > sc.png
复制代码
```

如果 adb 版本较老，无法使用 `exec-out` 命令，这时候建议更新 adb 版本。无法更新的话可以使用以下麻烦点的办法：

先截图保存到设备里：

```
adb shell screencap -p /sdcard/sc.png
复制代码
```

然后将 png 文件导出到电脑：

```
adb pull /sdcard/sc.png
复制代码
```

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。

另外一种一行命令截图并保存到电脑的方法： *Linux 和 Windows*

```
adb shell screencap -p | sed "s/\r$//" > sc.png
```
复制代码

Mac OS X

```
adb shell screencap -p | gsed "s/\r$//" > sc.png
```
复制代码

这个方法需要用到 gnu sed 命令，在 Linux 下直接就有，在 Windows 下 Git 安装目录的 bin 文件夹下也有。如果确实找不到该命令，可以下载 sed for Windows 并将 sed.exe 所在文件夹添加到 PATH 环境变量里。

而在 Mac 下使用系统自带的 sed 命令会报错：

```
sed: RE error: illegal byte sequence
```
复制代码

需要安装 gnu-sed，然后使用 gsed 命令：

```
brew install gnu-sed
```
复制代码

4.11.2 录制屏幕

录制屏幕以 mp4 格式保存到 /sdcard：

```
adb shell screenrecord /sdcard/filename.mp4
```
复制代码

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
```
复制代码

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。

4.11.3 查看连接过的 WiFi 密码

注：需要 root 权限。

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

4.11.4 设置系统日期和时间

注：需要 root 权限。

```
adb shell
su
date -s 20160823.131500
复制代码
```

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。

4.11.5 重启手机

```
adb reboot
复制代码
```

4.11.6 检测设备是否已 root

```
adb shell
su
复制代码
```

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

4.11.7 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
复制代码
```

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 。

4.11.8 开启 / 关闭 WiFi

注：需要 root 权限。

- 开启 WiFi：

```
adb root
adb shell svc wifi enable
复制代码
```

- 关闭 WiFi：

```
adb root
adb shell svc wifi disable
复制代码
```

若执行成功，输出为空；若未取得 root 权限执行此命令，将执行失败，输出 `Killed` 。

4.12 刷机相关命令

4.12.1 重启到 Recovery 模式

```
adb reboot recovery
复制代码
```

4.12.2 从 Recovery 重启到 Android

```
adb reboot
复制代码
```

4.12.2 重启到 Fastboot 模式

```
adb reboot bootloader
```
复制代码

4.12.4 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。

```
adb reboot recovery
```
复制代码

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。

```
adb sideload <path-to-update.zip>
```
复制代码

4.13 安全相关命令

4.13.1 启用 / 禁用 SELinux

启用 SELinux

```
adb root
adb shell setenforce 1
```
复制代码

禁用 SELinux

```
adb root
adb shell setenforce 0
```
复制代码

4.13.2 启用 / 禁用 dm_verity

启用 dm_verity

```
adb root
adb enable-verity
```

复制代码

禁用 dm_verity

```
adb root
adb disable-verity
```
复制代码

4.14 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。

4.14.1 查看进程状态

```
adb shell ps
```
复制代码

输出信息各列含义：

列名	含义
USER	所属用户
PID	进程 ID
PPID	父进程 ID
NAME	进程名

4.14.2 查看处理器实时状态

```
adb shell top [-m max_procs] [-n iterations] [-d delay] [-s sort_column
```
复制代码

`adb shell top` 后面可以跟一些可选参数进行过滤查看不同的列表，可用参数及含义如下：

参数	含义
-m	最多显示多少个进程
-n	刷新多少次后退出
-d	刷新时间间隔 (单位秒，默认值 5)

参数	含义
-s	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

输出信息各列含义：

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

4.14.3 查看进程 UID

有两种方案：

1. `adb shell dumpsys package <packagename> | grep userId=`

   如：

```
adb shell dumpsys package org.mazhuang.guanggoo | grep userId=
  userId=10394
```

复制代码

2. 通过 ps 命令找到对应进程的 pid 之后 `adb shell cat /proc/<pid>/status | grep Uid` 如：

```
adb shell
gemini:/ $ ps | grep org.mazhuang.guanggoo
u0_a394   28635 770   1795812 78736 SyS_epoll_ 0000000000 S org
gemini:/ $ cat /proc/28635/status | grep Uid
Uid:    10394   10394   10394   10394
gemini:/ $
```

复制代码

五、致谢

1. ADB Shell
2. Awesome Adb
3. Android Debug Bridge

全文完