# Android adb命令大全

□Authored by XuPeng

ADB，即 Android Debug Bridge，它是 Android 开发/测试人员不可替代的强大工具，也是 Android 设备玩家的好玩具。

**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。

# [基本用法](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%9F%BA%E6%9C%AC%E7%94%A8%E6%B3%95)

* + [命令语法](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%91%BD%E4%BB%A4%E8%AF%AD%E6%B3%95)
  + [为命令指定目标设备](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%B8%BA%E5%91%BD%E4%BB%A4%E6%8C%87%E5%AE%9A%E7%9B%AE%E6%A0%87%E8%AE%BE%E5%A4%87)
  + [启动/停止](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%90%AF%E5%8A%A8%E5%81%9C%E6%AD%A2)
  + [查看 adb 版本](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B-adb-%E7%89%88%E6%9C%AC)
  + [以 root 权限运行 adbd](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%BB%A5-root-%E6%9D%83%E9%99%90%E8%BF%90%E8%A1%8C-adbd)
  + [指定 adb server 的网络端口](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%8C%87%E5%AE%9A-adb-server-%E7%9A%84%E7%BD%91%E7%BB%9C%E7%AB%AF%E5%8F%A3)
* [设备连接管理](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%AE%BE%E5%A4%87%E8%BF%9E%E6%8E%A5%E7%AE%A1%E7%90%86)
  + [查询已连接设备/模拟器](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E8%AF%A2%E5%B7%B2%E8%BF%9E%E6%8E%A5%E8%AE%BE%E5%A4%87%E6%A8%A1%E6%8B%9F%E5%99%A8)
  + [USB 连接](https://github.com/mzlogin/awesome-adb/blob/master/README.md#usb-%E8%BF%9E%E6%8E%A5)
  + [无线连接（需要借助 USB 线）](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%97%A0%E7%BA%BF%E8%BF%9E%E6%8E%A5%E9%9C%80%E8%A6%81%E5%80%9F%E5%8A%A9-usb-%E7%BA%BF)
  + [无线连接（无需借助 USB 线）](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%97%A0%E7%BA%BF%E8%BF%9E%E6%8E%A5%E6%97%A0%E9%9C%80%E5%80%9F%E5%8A%A9-usb-%E7%BA%BF)
* [应用管理](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%BA%94%E7%94%A8%E7%AE%A1%E7%90%86)
  + [查看应用列表](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E5%BA%94%E7%94%A8%E5%88%97%E8%A1%A8)
    - [所有应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%89%80%E6%9C%89%E5%BA%94%E7%94%A8)
    - [系统应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%B3%BB%E7%BB%9F%E5%BA%94%E7%94%A8)
    - [第三方应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%AC%AC%E4%B8%89%E6%96%B9%E5%BA%94%E7%94%A8)
    - [包名包含某字符串的应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%8C%85%E5%90%8D%E5%8C%85%E5%90%AB%E6%9F%90%E5%AD%97%E7%AC%A6%E4%B8%B2%E7%9A%84%E5%BA%94%E7%94%A8)
  + [安装 APK](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%AE%89%E8%A3%85-apk)
  + [卸载应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%8D%B8%E8%BD%BD%E5%BA%94%E7%94%A8)
  + [清除应用数据与缓存](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%B8%85%E9%99%A4%E5%BA%94%E7%94%A8%E6%95%B0%E6%8D%AE%E4%B8%8E%E7%BC%93%E5%AD%98)
  + [查看前台 Activity](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E5%89%8D%E5%8F%B0-activity)
  + [查看正在运行的 Services](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E6%AD%A3%E5%9C%A8%E8%BF%90%E8%A1%8C%E7%9A%84-services)
* [与应用交互](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%B8%8E%E5%BA%94%E7%94%A8%E4%BA%A4%E4%BA%92)
  + [调起 Activity](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%B0%83%E8%B5%B7-activity)
  + [调起 Service](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%B0%83%E8%B5%B7-service)
  + [发送广播](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%8F%91%E9%80%81%E5%B9%BF%E6%92%AD)
  + [强制停止应用](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%BC%BA%E5%88%B6%E5%81%9C%E6%AD%A2%E5%BA%94%E7%94%A8)
* [文件管理](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%96%87%E4%BB%B6%E7%AE%A1%E7%90%86)
  + [复制设备里的文件到电脑](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%A4%8D%E5%88%B6%E8%AE%BE%E5%A4%87%E9%87%8C%E7%9A%84%E6%96%87%E4%BB%B6%E5%88%B0%E7%94%B5%E8%84%91)
  + [复制电脑里的文件到设备](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%A4%8D%E5%88%B6%E7%94%B5%E8%84%91%E9%87%8C%E7%9A%84%E6%96%87%E4%BB%B6%E5%88%B0%E8%AE%BE%E5%A4%87)
* [模拟按键/输入](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%A8%A1%E6%8B%9F%E6%8C%89%E9%94%AE%E8%BE%93%E5%85%A5)
  + [电源键](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%94%B5%E6%BA%90%E9%94%AE)
  + [菜单键](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%8F%9C%E5%8D%95%E9%94%AE)
  + [HOME 键](https://github.com/mzlogin/awesome-adb/blob/master/README.md#home-%E9%94%AE)
  + [返回键](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%BF%94%E5%9B%9E%E9%94%AE)
  + [音量控制](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%9F%B3%E9%87%8F%E6%8E%A7%E5%88%B6)
  + [媒体控制](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%AA%92%E4%BD%93%E6%8E%A7%E5%88%B6)
  + [点亮/熄灭屏幕](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%82%B9%E4%BA%AE%E7%86%84%E7%81%AD%E5%B1%8F%E5%B9%95)
  + [滑动解锁](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%BB%91%E5%8A%A8%E8%A7%A3%E9%94%81)
  + [输入文本](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%BE%93%E5%85%A5%E6%96%87%E6%9C%AC)
* [查看日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E6%97%A5%E5%BF%97)
  + [Android 日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#android-%E6%97%A5%E5%BF%97)
    - [按级别过滤日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%8C%89%E7%BA%A7%E5%88%AB%E8%BF%87%E6%BB%A4%E6%97%A5%E5%BF%97)
    - [按 tag 和级别过滤日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%8C%89-tag-%E5%92%8C%E7%BA%A7%E5%88%AB%E8%BF%87%E6%BB%A4%E6%97%A5%E5%BF%97)
    - [日志格式](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%97%A5%E5%BF%97%E6%A0%BC%E5%BC%8F)
    - [清空日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%B8%85%E7%A9%BA%E6%97%A5%E5%BF%97)
  + [内核日志](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%86%85%E6%A0%B8%E6%97%A5%E5%BF%97)
* [查看设备信息](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E8%AE%BE%E5%A4%87%E4%BF%A1%E6%81%AF)
  + [型号](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%9E%8B%E5%8F%B7)
  + [电池状况](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%94%B5%E6%B1%A0%E7%8A%B6%E5%86%B5)
  + [屏幕分辨率](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%B1%8F%E5%B9%95%E5%88%86%E8%BE%A8%E7%8E%87)
  + [屏幕密度](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%B1%8F%E5%B9%95%E5%AF%86%E5%BA%A6)
  + [显示屏参数](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%98%BE%E7%A4%BA%E5%B1%8F%E5%8F%82%E6%95%B0)
  + [android\_id](https://github.com/mzlogin/awesome-adb/blob/master/README.md#android_id)
  + [IMEI](https://github.com/mzlogin/awesome-adb/blob/master/README.md#imei)
  + [Android 系统版本](https://github.com/mzlogin/awesome-adb/blob/master/README.md#android-%E7%B3%BB%E7%BB%9F%E7%89%88%E6%9C%AC)
  + [IP 地址](https://github.com/mzlogin/awesome-adb/blob/master/README.md#ip-%E5%9C%B0%E5%9D%80)
  + [Mac 地址](https://github.com/mzlogin/awesome-adb/blob/master/README.md#mac-%E5%9C%B0%E5%9D%80)
  + [CPU 信息](https://github.com/mzlogin/awesome-adb/blob/master/README.md#cpu-%E4%BF%A1%E6%81%AF)
  + [内存信息](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%86%85%E5%AD%98%E4%BF%A1%E6%81%AF)
  + [更多硬件与系统属性](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9B%B4%E5%A4%9A%E7%A1%AC%E4%BB%B6%E4%B8%8E%E7%B3%BB%E7%BB%9F%E5%B1%9E%E6%80%A7)
* [修改设置](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%BF%AE%E6%94%B9%E8%AE%BE%E7%BD%AE)
  + [分辨率](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%88%86%E8%BE%A8%E7%8E%87)
  + [屏幕密度](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%B1%8F%E5%B9%95%E5%AF%86%E5%BA%A6-1)
  + [显示区域](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%98%BE%E7%A4%BA%E5%8C%BA%E5%9F%9F)
  + [关闭 USB 调试模式](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%85%B3%E9%97%AD-usb-%E8%B0%83%E8%AF%95%E6%A8%A1%E5%BC%8F)
  + [状态栏和导航栏的显示隐藏](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E7%8A%B6%E6%80%81%E6%A0%8F%E5%92%8C%E5%AF%BC%E8%88%AA%E6%A0%8F%E7%9A%84%E6%98%BE%E7%A4%BA%E9%9A%90%E8%97%8F)
* [实用功能](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%AE%9E%E7%94%A8%E5%8A%9F%E8%83%BD)
  + [屏幕截图](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%B1%8F%E5%B9%95%E6%88%AA%E5%9B%BE)
  + [录制屏幕](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%BD%95%E5%88%B6%E5%B1%8F%E5%B9%95)
  + [重新挂载 system 分区为可写](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%87%8D%E6%96%B0%E6%8C%82%E8%BD%BD-system-%E5%88%86%E5%8C%BA%E4%B8%BA%E5%8F%AF%E5%86%99)
  + [查看连接过的 WiFi 密码](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E8%BF%9E%E6%8E%A5%E8%BF%87%E7%9A%84-wifi-%E5%AF%86%E7%A0%81)
  + [设置系统日期和时间](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%AE%BE%E7%BD%AE%E7%B3%BB%E7%BB%9F%E6%97%A5%E6%9C%9F%E5%92%8C%E6%97%B6%E9%97%B4)
  + [重启手机](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%87%8D%E5%90%AF%E6%89%8B%E6%9C%BA)
  + [检测设备是否已 root](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%A3%80%E6%B5%8B%E8%AE%BE%E5%A4%87%E6%98%AF%E5%90%A6%E5%B7%B2-root)
  + [使用 Monkey 进行压力测试](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%BD%BF%E7%94%A8-monkey-%E8%BF%9B%E8%A1%8C%E5%8E%8B%E5%8A%9B%E6%B5%8B%E8%AF%95)
  + [开启/关闭 WiFi](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%BC%80%E5%90%AF%E5%85%B3%E9%97%AD-wifi)
* [刷机相关命令](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%88%B7%E6%9C%BA%E7%9B%B8%E5%85%B3%E5%91%BD%E4%BB%A4)
  + [重启到 Recovery 模式](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%87%8D%E5%90%AF%E5%88%B0-recovery-%E6%A8%A1%E5%BC%8F)
  + [从 Recovery 重启到 Android](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E4%BB%8E-recovery-%E9%87%8D%E5%90%AF%E5%88%B0-android)
  + [重启到 Fastboot 模式](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%87%8D%E5%90%AF%E5%88%B0-fastboot-%E6%A8%A1%E5%BC%8F)
  + [通过 sideload 更新系统](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E9%80%9A%E8%BF%87-sideload-%E6%9B%B4%E6%96%B0%E7%B3%BB%E7%BB%9F)
* [更多 adb shell 命令](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9B%B4%E5%A4%9A-adb-shell-%E5%91%BD%E4%BB%A4)
  + [查看进程](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E8%BF%9B%E7%A8%8B)
  + [查看实时资源占用情况](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E6%9F%A5%E7%9C%8B%E5%AE%9E%E6%97%B6%E8%B5%84%E6%BA%90%E5%8D%A0%E7%94%A8%E6%83%85%E5%86%B5)
  + [其它](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%85%B6%E5%AE%83)
* [常见问题](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%B8%B8%E8%A7%81%E9%97%AE%E9%A2%98)
  + [启动 adb server 失败](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%90%AF%E5%8A%A8-adb-server-%E5%A4%B1%E8%B4%A5)
* [adb 的非官方实现](https://github.com/mzlogin/awesome-adb/blob/master/README.md#adb-%E7%9A%84%E9%9D%9E%E5%AE%98%E6%96%B9%E5%AE%9E%E7%8E%B0)
* [致谢](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E8%87%B4%E8%B0%A2)
* [参考链接](https://github.com/mzlogin/awesome-adb/blob/master/README.md#%E5%8F%82%E8%80%83%E9%93%BE%E6%8E%A5)

**基本用法**

**命令语法格式**

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 10.129.164.6:5555 device

C:\Users\Administrator>adb devices

List of devices attached

c876718 device

127.0.0.1:62001 device

输出里的 cf264b8f、emulator-5554 和 10.129.164.6:5555 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

C:\Users\Administrator>adb -d shell wm size

Physical size: 1080x1920

C:\Users\Administrator>adb -s c876718 shell wm size

Physical size: 1080x1920

又如想给 10.129.164.6:5555 这个设备安装应用（*这种形式的 serialNumber 格式为 <IP>:<Port>，一般为无线连接的设备或 Genymotion 等第三方 Android 模拟器*）：

adb -s 10.129.164.6:5555 install test.apk

**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。**

**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 1.0.31

**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。

**设备连接管理**

**连接夜神安卓模拟器**

命令：

adb connect 127.0.0.1:62001

输出示例：

connected to 127.0.0.1:62001

**查询已连接设备/模拟器**

命令：

adb devices

输出示例：

C:\Users\Administrator>adb devices

List of devices attached

c876718 device

127.0.0.1:62001 device

emulator-5554 device

输出格式为： [serialNumber] [state]，serialNumber （序列号）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三台模拟器/安卓设备，c876718，emulator-5554和127.0.0.1:62001 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，而 127.0.0.1:62001 这种形为 <IP>:<Port> 的 serialNumber 一般是无线连接的设备或 Genymotion 等第三方 Android 模拟器。

常见异常输出：

1. 没有设备/模拟器连接成功。

List of devices attached

1. 设备/模拟器未连接到 adb 或无响应。

List of devices attached 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。

1. 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，那需要通过相关操作让它显示出来：在「设置」-「关于手机」连续点击「版本号」7 次。

1. 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。

1. 通过 USB 线连接好电脑和设备后确认状态。

adb devices

如果能看到

xxxxxx device

说明连接成功。

**无线连接（需要借助 USB 线）**

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！

操作步骤：

1. 将 Android 设备与要运行 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）。

1. 让设备在 5555 端口监听 TCP/IP 连接：

adb tcpip 5555

1. 断开 USB 连接。
2. 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，也可以使用下文里 [查看设备信息 - IP 地址](https://github.com/mzlogin/awesome-adb/blob/master/README.md#ip-%E5%9C%B0%E5%9D%80) 一节里的方法用 adb 命令来查看。

1. 通过 IP 地址连接设备。

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。

1. 确认连接状态。

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>

**无线连接（无需借助 USB 线）**

**注：需要 root 权限。**

既然我们已经实现无线连接，那能不能所有步骤下来都是无线的呢？答案是肯定的。

1. 在 Android 设备上安装一个终端模拟器。

已经安装过的设备可以跳过此步。我使用的终端模拟器下载地址是：[Terminal Emulator for Android Downloads](https://jackpal.github.io/Android-Terminal-Emulator/)

1. 将 Android 设备与要运行 adb 的电脑连接到同一个局域网，比如连到同一个 WiFi。
2. 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

1. 找到 Android 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，也可以使用下文里 [查看设备信息 - IP 地址](https://github.com/mzlogin/awesome-adb/blob/master/README.md#ip-%E5%9C%B0%E5%9D%80) 一节里的方法用 adb 命令来查看。

1. 在电脑上通过 adb 和 IP 地址连接 Android 设备。

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。

如果能看到 connected to <device-ip-address>:5555 这样的输出则表示连接成功。

*节注一：*

有的设备，比如小米 5S + MIUI 8.0 + Android 6.0.1 MXB48T，可能在第 5 步之前需要重启 adbd 服务，在设备的终端模拟器上运行：

restart adbd

如果 restart 无效，尝试以下命令：

start adbd stop adbd

**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 |
| -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 –d shell pm list packages

或者

adb -s c876718 shell pm list packages # c876718为物理设备的SN

输出示例：

package:com.miui.screenrecorder

package:com.miui.contentextension

package:com.android.providers.telephony

package:com.miui.powerkeeper

package:com.unionpay.tsmservice.mi

package:com.miui.fm

package:com.supercell.clashofclans.mi

package:com.android.providers.calendar

package:com.miui.contentcatcher

package:com.xiaomi.vipaccount

package:com.android.providers.media

package:com.milink.service

package:com.qiyi.video

**系统应用**

命令：

adb –d shell pm list packages –s

或者

adb -s c876718 shell pm list packages –s

**第三方应用**

命令：

adb –d shell pm list packages -3

或者

adb -s c876718 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 taobao 的应用列表，命令：

adb -d shell pm list packages taobao

当然也可以使用 grep 来过滤：

C:\Users\Administrator>adb -d shell pm list packages | grep taobao

'grep' 不是内部或外部命令，也不是可运行的程序

或批处理文件。

C:\Users\Administrator>adb -d shell "pm list packages | grep taobao"

package:com.taobao.taobao

**# 建议凡是出现shell后面的内容，都叫英文的双引号“”””!**

**安装 APK【待审查】**

命令格式：

adb install [-lrtsdg] <path\_to\_apk>

参数：

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 |
| -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 |

运行命令后如果见到类似如下输出（状态为 Success）代表安装成功：

[100%] /data/local/tmp/1.apk pkg: /data/local/tmp/1.apk Success

上面是当前最新版 v1.0.36 的 adb 的输出，会显示 push apk 文件到手机的进度百分比。

使用旧版本 adb 的输出则是这样的：

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，比如：

[100%] /data/local/tmp/map-20160831.apk pkg: /data/local/tmp/map-20160831.apk Failure [INSTALL\_FAILED\_ALREADY\_EXISTS]

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，或卸载了但没卸载干净 | adb install 时使用 -r 参数，或者先 adb uninstall <packagename> 再安装 |
| 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 | 以前安装过同名应用，但卸载时数据没有移除；或者已安装该应用，但签名不一致 | 先 adb uninstall <packagename> 再安装 |
| 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 |  |
| 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 | 1. sdcard 访问失败; 2. 应用签名与 ROM 签名一致，被当作内置应用 | 1. 确认 sdcard 可用，或者安装到内置存储; 2. 打包时不与 ROM 使用相同签名 |
| INSTALL\_FAILED\_INVALID\_INSTALL\_LOCATION | 1. 不能安装到指定位置; 2. 应用签名与 ROM 签名一致，被当作内置应用 | 1. 切换安装位置，添加或删除 -s 参数; 2. 打包时不与 ROM 使用相同签名 |
| 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 |
| 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 |  |
| 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 device | 空间不足 | 清理空间 |
| 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致 | 先卸载设备上的该应用，再安装 |

参考：[PackageManager.java](https://github.com/android/platform_frameworks_base/blob/master/core%2Fjava%2Fandroid%2Fcontent%2Fpm%2FPackageManager.java)

*adb install 内部原理简介*

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。

所以，必要的时候也可以根据这个步骤，手动分步执行安装过程。

**卸载应用【待补充】**

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【待补充】**

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：

Linux

adb shell dumpsys activity activities | grep mFocusedActivity

windows

adb –d shell dumpsys activity | findstr "mFocusedActivity"

输出示例：

mFocusedActivity: ActivityRecord{9c8c310 u0 com.wandoujia.phoenix2/com.pp.assistant.activity.PPMainActivity t250}

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

**查看正在运行的 Services**

命令：

adb shell dumpsys activity services [<packagename>]

<packagename> 参数不是必须的，指定 <packagename> 表示查看与某个包名相关的 Services，不指定表示查看所有 Services。

<packagename> 不一定要给出完整的包名，比如运行 adb shell dumpsys activity services tencent，那么包名 tencent1、tencent2 和 tencent123 等相关的 Services 都会列出来。

**与应用交互【难度】？？？？？**

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>` |
| --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 |
| --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.AccountAuthenticatorService

表示调起微信的某 Service。

**发送广播**

命令格式：

adb shell am broadcast [options] <INTENT>

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

又例如，只向 org.mazhuang.boottimemeasure/.BootCompletedReceiver 广播 BOOT\_COMPLETED：

adb shell am broadcast -a android.intent.action.BOOT\_COMPLETED -n org.mazhuang.boottimemeasure/.BootCompletedReceiver

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 |
| 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）*

**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。

**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。

**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：

C:\Users\Administrator>adb shell

shell@cancro:/ $ input

input

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

The sources are:

mouse

keyboard

joystick

touchnavigation

touchpad

trackball

stylus

dpad

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](https://developer.android.com/reference/android/view/KeyEvent.html)，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 |
| 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。

**电源键**

命令：

C:\Users\Administrator>adb shell input keyevent 26

预期输出：

执行结果相当于按电源键

**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）：

C:\Users\Administrator>adb shell input keyevent 224

# 先点亮屏幕

C:\Users\Administrator>adb shell input swipe 300 1000 300 500

# 解锁

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【待细化】**

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。

**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优先级（priority）：

* 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。

（**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。）

**按 tag 和级别过滤日志**

<filter-spec> 可以由多个 <tag>[:priority] 组成。

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。

**查看设备信息**

**型号**

命令：

C:\Users\Administrator>adb shell getprop ro.product.model

输出示例：

MI 3W

**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

Max charging current: 500000

status: 5

health: 2

present: true

level: 100

scale: 100

voltage: 4308

temperature: 277

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 100% 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。

如果使用命令修改过，那输出可能是：

Physical size: 1080x1920 Override size: 480x1024

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

**屏幕密度**

命令：

adb shell wm density

输出示例：

Physical density: 480

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

表明设备的屏幕密度原来是 480dpi，当前被修改为 160dpi。

**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：

67688f1c1e159930

**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](http://stackoverflow.com/questions/27002663/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：

6.0.1

**IP 地址**

每次想知道设备的 IP 地址的时候都得「设置」-「关于手机」-「状态信息」-「IP地址」很烦对不对？通过 adb 可以方便地查看。

命令：

adb shell " ifconfig | grep Mask"

输出示例：

inet addr:127.0.0.1 Mask:255.0.0.0

inet addr:192.168.2.104 Bcast:192.168.2.255 Mask:255.255.255.0

那么 127.0.0.1 就是设备 IP 地址。

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

输出示例：

wlan0: ip 10.129.160.99 mask 255.255.240.0 flags [up broadcast running multicast]

wlan0 Link encap:UNSPEC

inet addr:192.168.2.104 Bcast:192.168.2.255 Mask:255.255.255.0

inet6 addr: fe80::d697:bff:fe4f:fc1f/64 Scope: Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:48143 errors:0 dropped:2 overruns:0 frame:0

TX packets:38716 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:65459602 TX bytes:3486035

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

输出示例：

wlan0 UP 10.129.160.99/20 0x00001043 f8:a9:d0:17:42:4d lo UP 127.0.0.1/8 0x00000049 00:00:00:00:00:00 p2p0 UP 0.0.0.0/0 0x00001003 fa:a9:d0:17:42:4d sit0 DOWN 0.0.0.0/0 0x00000080 00:00:00:00:00:00 rmnet0 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet1 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet3 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet2 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet4 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet6 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet5 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rmnet7 DOWN 0.0.0.0/0 0x00000000 00:00:00:00:00:00 rev\_rmnet3 DOWN 0.0.0.0/0 0x00001002 4e:b7:e4:2e:17:58 rev\_rmnet2 DOWN 0.0.0.0/0 0x00001002 4e:f0:c8:bf:7a:cf rev\_rmnet4 DOWN 0.0.0.0/0 0x00001002 a6:c0:3b:6b:c4:1f rev\_rmnet6 DOWN 0.0.0.0/0 0x00001002 66:bb:5d:64:2e:e9 rev\_rmnet5 DOWN 0.0.0.0/0 0x00001002 0e:1b:eb:b9:23:a0 rev\_rmnet7 DOWN 0.0.0.0/0 0x00001002 7a:d9:f6:81:40:5a rev\_rmnet8 DOWN 0.0.0.0/0 0x00001002 4e:e2:a9:bb:d0:1b rev\_rmnet0 DOWN 0.0.0.0/0 0x00001002 fe:65:d0:ca:82:a9 rev\_rmnet1 DOWN 0.0.0.0/0 0x00001002 da:d8:e8:4f:2e:fe

可以看到网络连接名称、启用状态、IP 地址和 Mac 地址等信息。

**Mac 地址**

命令：

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

输出示例：

d4:97:0b:4f:fc:1f

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。

**CPU 信息**

命令：

adb shell cat /proc/cpuinfo

输出示例：

Processor : ARMv7 Processor rev 1 (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 : 1

Hardware : Qualcomm MSM8974PRO-AB

Revision : 0000

Serial : 0000000000000000

这是 MI3 w 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 1 (v71)。

**内存信息**

命令：

adb shell cat /proc/meminfo

输出示例：

MemTotal: 1893504 kB

MemFree: 143328 kB

Buffers: 58548 kB

Cached: 587304 kB

SwapCached: 5116 kB

Active: 560644 kB

Inactive: 663164 kB

Active(anon): 297716 kB

Inactive(anon): 370704 kB

Active(file): 262928 kB

Inactive(file): 292460 kB

Unevictable: 86360 kB

Mlocked: 83560 kB

HighTotal: 1163260 kB

HighFree: 26396 kB

LowTotal: 730244 kB

LowFree: 116932 kB

SwapTotal: 786428 kB

SwapFree: 704584 kB

Dirty: 0 kB

Writeback: 0 kB

AnonPages: 662732 kB

Mapped: 238656 kB

Shmem: 4104 kB

Slab: 87812 kB

SReclaimable: 45820 kB

SUnreclaim: 41992 kB

KernelStack: 12648 kB

PageTables: 26772 kB

NFS\_Unstable: 0 kB

Bounce: 0 kB

WritebackTmp: 0 kB

CommitLimit: 1733180 kB

Committed\_AS: 27452656 kB

VmallocTotal: 122880 kB

VmallocUsed: 13380 kB

VmallocChunk: 75632 kB

其中，MemTotal 就是设备的总内存，MemFree 是当前空闲内存。

**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[*节注一*] |
| 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 |

*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"

示例输出：

# ro.product.cpu.abi and ro.product.cpu.abi2 are obsolete,

# use ro.product.cpu.abilist instead.

ro.product.cpu.abi=armeabi-v7a

ro.product.cpu.abi2=armeabi

ro.product.cpu.abilist=armeabi-v7a,armeabi

ro.product.cpu.abilist32=armeabi-v7a,armeabi

ro.product.cpu.abilist64=

**修改设置【待修改】**

**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。

**分辨率**

命令：

adb shell wm size 480x1024

表示将分辨率修改为 480px \* 1024px。

恢复原分辨率命令：

adb shell wm size reset

**屏幕密度**

命令：

adb shell wm density 160

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

**显示区域**

命令：

adb shell wm overscan 0,0,0,200

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

**关闭 USB 调试模式**

命令：

adb shell settings put global adb\_enabled 0

恢复：

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」。

**状态栏和导航栏的显示隐藏**

本节所说的相关设置对应 Cyanogenmod 里的「扩展桌面」。

命令：

adb shell settings put global policy\_control <key-values>

<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 |
| packagename | 指定应用 |
| -packagename | 排除指定应用 |

例如：

adb shell settings put global policy\_control immersive.full=\*

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.package1,com.package2:immersive.navigation=apps,-com.package3

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。

**实用功能【待修改】**

**屏幕截图**

截图保存到电脑：

adb exec-out screencap -p > sc.png

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

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。

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

*Mac OS X*

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

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](http://gnuwin32.sourceforge.net/packages/sed.htm) 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

需要安装 gnu-sed，然后使用 gsed 命令：

brew install gnu-sed

**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

1. 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=1000,gid=1000,fmask=0337,dmask=0227,codepage=cp437,iocharset=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,default\_permissions,allow\_other 0 0 /dev/fuse /mnt/shell/emulated/0 fuse rw,nosuid,nodev,relatime,user\_id=1023,group\_id=1023,default\_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

1. 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](https://developer.android.com/studio/test/monkey.html)。

**开启/关闭 WiFi**

**注：需要 root 权限。**

有时需要控制设备的 WiFi 状态，可以用以下指令完成。

开启 WiFi：

adb root adb shell svc wifi enable

关闭 WiFi：

adb root adb shell svc wifi disable

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。

**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

1. 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。

1. 通过 adb 上传和更新系统。

命令：

adb sideload <path-to-update.zip>

**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 |
| 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 |
| top | 查看进程的资源占用情况 |

**常见问题**

**启动 adb server 失败**

**出错提示**

error: protocol fault (couldn't read status): No error

**可能原因**

adb server 进程想使用的 5037 端口被占用。

**解决方案**

找到占用 5037 端口的进程，然后终止它。以 Windows 下为例：

netstat -ano | findstr LISTENING ... TCP 0.0.0.0:5037 0.0.0.0:0 LISTENING 1548 ...

这里 1548 即为进程 ID，用命令结束该进程：

taskkill /PID 1548

然后再启动 adb 就没问题了。