adb

译者署名： 移动云_文斌
译者链接：http://blog.csdn.net/caowenbin
版本：Android 2.3 r1

Android Debug Bridge

Android 调试桥接器，简称 adb，是用于管理模拟器或真机状态的万能工具，采用了客户端-服务器模型，包括三个部分：

- 客户端部分，运行在开发用的电脑上，可以在命令行中运行 adb 命令来调用该客户端，像 ADB 插件和 DDMS 这样的 Android 工具也可以调用 adb 客户端。
- 服务端部分，是运行在开发用电脑上的后台进程，用于管理客户端与运行在模拟器或真机的守护进程通信。
- 守护进程部分，运行于模拟器或手机的后台。

当启动 adb 客户端时，客户端首先检测 adb 服务端进程是否运行，如果没有运行，则启动服务端。当服务端启动时，它会绑定到本地的 TCP5037 端口，并且监听从 adb 客户端发来的命令——所有的 adb 客户端都使用 5037 端口与 adb 服务端通信。

接下来服务端与所有正在运行的模拟器或手机连接。它通过扫描 5555-5585 之间的奇数号端口来搜索模拟器或手机，一旦发现 adb 守护进程，就通过此端口进行连接。需要说明的是，每一个模拟器或手机使用一对有序的端口，偶数号端口用于控制台连接，奇数号端口用于 adb 连接，例如：

Emulator 1, console: 5554
Emulator 1, adb: 5555
Emulator 2, console: 5556
Emulator 2, adb: 5557 ...

即如果模拟器与 adb 在 5555 端口连接，则其与控制台的连接就是 5554 端口。

当服务端与所有的模拟器建立连接之后，就可以使用 adb 命令来控制或者访问了。因为服务端管理着连接并且可以接收到从多个 adb 客户端的命令，所以可以从任何一个客户端或脚本来控制任何模拟器或手机设备。

下文介绍了可以用来管理模拟器或手机的这些 adb 命令。如果是在 Eclipse 并且安装了 ADT 插件的环境下开发 Android 应用程序，就不需要从命令行使用 adb 了，ADT 插件已经提供了透明的集成。不过，还是可以在调试等需要的时候直接使用 adb。

使用 adb 命令

从开发用电脑的命令行或脚本文件中使用 adb 命令的用法是：

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

当使用的时候，程序会调用 adb 客户端。因为 adb 客户端不需要关联到任何模拟器，所以如果有多个模拟器或手机正在运行，就需要使用-d 参数指定要操作的是哪一个，更多关于这些选项参数的使用可以参见 Directing Commands to a Specific Emulator/Device Instance。

查询模拟器或手机状态

了解 adb 服务端连接的模拟器或手机可以帮助更好的使用 adb 命令，这可以通过 devices

命令列举出来：

```
adb devices
```

执行结果是 adb 为每一个设备输出以下状态信息：

- 序列号(serialNumber) — 由 adb 创建的使用控制台端口号的用于唯一标识一个模拟器或手机设备的字符串，格式是 <设备类型>-<端口号>，例如： emulator-5554
- 状态(state) — 连接状态，其值是：
  - offline — 未连接或未响应
  - device —已经连接到服务商。注意这个状态并不表示 Android 系统已经完全启动起来，系统启动的过程中已经可以连接 adb，但这个状态是正常的可操作状态。

每一个设备的输出形如：

```
[serialNumber] [state]
```

下面是 devices 命令和其执行结果：

```
$ adb devices
List of devices attached
emulator-5554  device
emulator-5556  device
emulator-5558  device
```

如果没有模拟器或手机在运行，该状态返回的是 no device。

操作指定的模拟器或手机

如果有多个模拟器或手机正在运行，当使用 adb 命令的时候就需要指定目标设备，这可以通过使用-s 选项参数实现，用法是：

```
adb -s <serialNumber> <command>
```

即可以在 adb 命令中使用序列号指定特定的目标，前文已经提到的 devices 命令可以实现查询设备的序列号信息。

例如：

```
adb -s emulator-5556 install helloWorld.apk
```

需要注意的是，如果使用了-s 而没有指定设备的话，adb 会报错。

安装应用程序

可以使用 adb 从开发用电脑中复制应用程序并且安装到模拟器或手机上，使用 install 命令即可，在这个命令中，必须指定待安装的.apk 文件的路径：

```
adb install <path_to_apk>
```

关于创建可安装的应用的更多信息，请参见 Android Asset Packaging Tool (aapt).

注意，如果使用了安装有 ADT 插件的 Eclipse 开发环境，就不需要直接使用 adb 或 aapt 命令来安装应用程序了，ADT 插件可以自动完成这些操作。

转发端口

可以使用 forward 命令转发端口 — 将特定端口上的请求转发到模拟器或手机的不同的端口上。下例是从 6100 端口转到 7100 端口：

```
adb forward tcp:6100 tcp:7100
```

也可以使用 UNIX 命名的 socket 标识：

```
adb forward tcp:6100 local:logd
```

与模拟器或手机传输文件

可以使用 adb 的 pull 和 push 命令从模拟器或手机中复制文件，或者将文件复制到模拟器或手机中。与 install 命令不同，它仅能复制.apk 文件到特定的位置， pull 和 push 命令可以复制任意文件夹和文件到模拟器或手机的任何位置。

从模拟器或手机中复制一个文件或文件夹（递归的）使用：

```
adb pull <remote> <local>
```

复制一个文件或文件夹（递归的）到模拟器或手机中使用：

```
adb push <local> <remote>
```

在这个命令中<local>和<remote>引用的是文件或文件夹的路径，在开发用电脑上的是 local，在模拟器或手机上的是 remote。

例如：

```
adb push foo.txt /sdcard/foo.txt
```

adb 命令列表

下表列出了所有 adb 支持的命令及其说明：

类别	命令	说明	备注
可选项	-d	命令仅对 USB 设备有效	如果有多个 USB 设备就会返回错误
	-e	命令仅对运行中的模拟器有效	如果有多个运行中的模拟器就会返回错误
	-s <serialNumber>	命令仅对 adb 关联的特定序列号的模拟器或手机有效(例如 "emulator-5556").	如果不指定设备就会返回错误
一般项	devices	输出所有关联的模拟器或手机设备列表	参见 Querying for Emulator/Device Instances 以获得更多信息。
	help	输出 adb 支持的命令	
	version	输出 adb 的版本号	
调试项	logcat [<option>] [<filter-specs>]	在屏幕上输出日志信息	

	bugreport	为报告 bug，在屏幕上输出 dumpsys，dumpstate 和 logcat 数据	
	jdwp	输出有效的 JDWP 进程信息	可以使用 forward jdwp:<pid> 转换端口以连接到指定的 JDWP 进程，例如： adb forward tcp:8000 jdwp:472 jdb -attach localhost:8000
数据项	install <path-to-apk>	安装应用程序（用完整路径指定.apk 文件）	
	pull <remote> <local>	从开发机 COPY 指定的文件到模拟器或手机	
	push <local> <remote>	从模拟器或手机 COPY 文件到开发机	
端口和网络项	forward <local> <remote>	从本地端口转换连接到模拟器或手机的指定端口	端口可以使用以下格式表示： ● tcp:<portnum> ● local:<UNIX domain socket name> ● dev:<character device name> ● jdwp:<pid>
	ppp <tty> [parm]...	通过 USB 运行 UPP ● <tty> —PPP 流中的 tty。例如:/dev/omap_csmi_ttyl。 ● [parm]... — 0 到多个 PPP/PPPD 选项，例如 defaultroute, local, notty 等等。 注意不用自动启动 PPP 连接	
脚本项	get-serialno	输出 adb 对象的序列号	参见 Querying for Emulator/Device Instances 以获得更多信息。
	get-state	输出 adb 设备的状态	
	wait-for-device	阻塞执行直到设备已经连接，即设备状态是 device.	可以在其他命令前加上此项，那样的话 adb 就会等到模拟器或手机设备已经连接才会执行命令，例如:

			`adb wait-for-device shell getprop`
			注意该命令并不等待系统完全启动，因此不能追加需要在系统完全启动才能执行的命令，例如 install 命令需要 Android 包管理器支持，但它必须在系统完全启动后才有效。下面的命令
			`adb wait-for-device install <app>.apk`
			会在模拟器或手机与 adb 发生连接后就执行 install，但系统还没有完全启动，所以会引起错误。

服务端项	start-server	检测 adb 服务进程是否启动，如果没启动则启动它。	
	kill-server	终止服务端进程	
Shell	shell	在目标模拟器或手机上启动远程 SHELL	参见 Issuing Shell Commands 以获得更多信息。
	shell [<shellCommand>]	在目标模拟器或手机上执行 shellCommand 然后退出远程 SHELL	

执行 Shell 命令

Adb 提供了 shell 来在模拟器或手机上运行各种各样的命令，这些命令的二进制形式存在于这个路径中：

```
/system/bin/...
```

无论是否进入 adb 远程 shell，都可以使用 shell 命令来执。

在未进入远程 shell 的情况下可以按下述格式执行单条命令：

```
adb [-d|-e|-s {<serialNumber>}] shell <shellCommand>
```

启动远程 shell 使用下面的格式：

```
adb [-d|-e|-s {<serialNumber>}] shell
```

退出远程 shell 时使用 CTRL+D 或 exit 终止会话。

以下是可以使用的 shell 命令的更多信息。

从远程 shell 检查 sqlite3 数据库

通过远程 shell，可以使用 sqlite3 命令行程序来管理由应用程序创建的 SQLite 数据库。sqlite3 工具包含很多有用的命令，例如 .dump 用于输出表格的内容，.schema 用于为已经存在的表输出 SQL CREATE 语句。 并且该工具也提供了联机执行 SQLite 命令的能力。

使用 sqlite3 时，向前文描述的那样进入模拟器的远程 shell，然后使用 sqlite3 命令。也

可以在调用 sqlite3 时指定数据库的全路径。SQLite3 数据库存储在 /data/data/<package_name>/databases/路径下。

示例:

```
$ adb -s emulator-5554 shell
# sqlite3
/data/data/com.example.google.rss.rssexample/databases/rssitems.db
SQLite version 3.3.12
Enter ".help" for instructions
.... enter commands, then quit...
sqlite> .exit
```

一旦运行了 sqlite3，就可以使用 sqlite3 命令，退出并返回远程 shell 可以使用 exit 或 CTRL+D。

使用 Monkey 进行 UI 或应用程序测试

Monkey 是运行于模拟器或手机上的一个程序，通过生成伪随机的大量的系统级的用户事件流来模拟操作，包括单击、触摸、手势等。从而为正在开发中的应用程序通过随机响应进行压力测试。

最简单使用 monkey 的方式是通过下面的命令行，它可以运行指定的应用程序并向其发送 500 个伪随机事件。

```
$ adb shell monkey -v -p your.package.name 500
```

关于 monkey 更多的选项及详细信息，请参见 UI/Application Exerciser Monkey。

其他 Shell 命令

下表列出了很多有效的 adb shell 命令，完整的列表可以通过启动模拟器并且使用 adb -help 命令获取。

```
adb shell ls /system/bin
```

帮助对于大部分命令是有效的。

Shell 命令	描述	备注
dumpsys	在屏幕上显示系统数据	The Dalvik Debug Monitor Service (DDMS) 工具提供了更易于使用的智能的调试环境。
dumpstate	将状态输出到文件	
logcat [<option>]... [<filter-spec>]...	输出日志信息	
dmesg	在屏幕上输出核心调试信息	
start	启动或重新启动模拟器或手机	
stop	停止模拟器或手机	

使用 logcat 查看日志

Android 日志系统提供了从众多应用程序和系统程序中收集和查看调试信息的机制，这些信息被收集到一系统循环缓冲区中，可以 logcat 命令查看和过滤。

使用 logcat 命令

查看和跟踪系统日志缓冲区的命令 logcat 的一般用法是：

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

下文介绍过滤器和命令选项，详细内容可参见 Listing of logcat Command Options。

可以在开发机中通过远程 shell 的方式使用 logcat 命令查看日志输出：

```
$ adb logcat
```

如果是在远程 shell 中可直接使用命令：

```
# logcat
```

过滤日志输出

每一条日志消息都有一个标记和优先级与其关联。

- 标记是一个简短的字符串，用于标识原始消息的来源 (例如"View" 来源于显示系统)。
- 优先级是下面的字符，顺序是从低到高：
  - V — 明细 (最低优先级)
  - D — 调试
  - I — 信息
  - W — 警告
  - E — 错误
  - F — 严重错误
  - S — 无记载 (最高优先级，没有什么会被记载)

通过运行 logcat ，可以获得一个系统中使用的标记和优先级的列表，观察列表的前两列，给出的格式是<priority>/<tag>。

这里是一个日志输出的消息，优先级是“I”，标记是“ActivityManager”：

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

如果想要减少输出的内容，可以加上过滤器表达式进行限制，过滤器可以限制系统只输出感兴趣的标记-优先级组合。

过滤器表达式的格式是 tag:priority ... ，其中 tag 是标记， priority 是最小的优先级， 该标记标识的所有大于等于指定优先级的消息被写入日志。也可以在一个过滤器表达式中提供多个这样的过滤，它们之间用空格隔开。

下面给出的例子是仅输出标记为“ActivityManager”并且优先级大于等于“Info”和标记为“MyApp”并且优先级大于等于“Debug”的日志：

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

上述表达式最后的 *:S 用于设置所有标记的日志优先级为 S，这样可以确保仅有标记为“View”（译者注：应该为 ActivityManager，原文可能是笔误）和“MyApp”的日志被输出，

使用 *:S 是可以确保输出符合指定的过滤器设置的一种推荐的方式，这样过滤器就成为了日志输出的“白名单”。

下面的表达是显示所有优先级大于等于“warning”的日志：

```
adb logcat *:W
```

如果在开发用电脑上运行 logcat （相对于运行运程 shell 而言)，也可以通过 ANDROID_LOG_TAGS 环境变量设置默认的过滤器表达式：

```
export ANDROID_LOG_TAGS="ActivityManager:I MyApp:D *:S"
```

需要注意的是，如果是在远程 shell 或是使用 adb shell logcat 命令运行 logcat ， ANDROID_LOG_TAGS 不会导出到模拟器或手机设备上。

控制日志格式

日志消息在标记和优先级之外还有很多元数据字段，这些字段可以通过修改输出格式来控制输出结果， -v 选项加上下面列出的内容可以控制输出字段：

- brief — 显示优先级/标记和原始进程的 PID (默认格式)
- process — 仅显示进程 PID
- tag — 仅显示优先级/标记
- thread — 仅显示进程：线程和优先级/标记
- raw — 显示原始的日志信息，没有其他的元数据字段
- time — 显示日期，调用时间，优先级/标记，PID
- long —显示所有的元数据字段并且用空行分隔消息内容

可以使用 -v 启动 logcat 来控制日志格式：

```
[adb] logcat [-v <format>]
```

例如使用 thread 输出格式：

```
adb logcat -v thread
```

注意只能在 -v 选项中指定一种格式。

Viewing Alternative Log Buffers

Android 日志系统为日志消息保持了多个循环缓冲区，而且不是所有的消息都被发送到默认缓冲区，要想查看这些附加的缓冲区，可以使用-b 选项，以下是可以指定的缓冲区：

- radio — 查看包含在无线/电话相关的缓冲区消息
- events — 查看事件相关的消息
- main — 查看主缓冲区 (默认缓冲区)

-b 选项的用法：

```
[adb] logcat [-b <buffer>]
```

例如查看 radio 缓冲区：

```
adb logcat -b radio
```

查看 stdout 和 stderr

默认的，Android 系统发送 stdout 和 stderr (System.out 和 System.err) 输出到 /dev/null。在 Dalvik VM 进程，可以将输出复制到日志文件，在这种情况下，系统使用 stdout 和 stderr 标记写入日志，优先级是 I。

要想使用这种方式获得输出，需要停止运行中的模拟器或手机，然后使用命令 setprop 来允许输出重定位，示例如下：

```
$ adb shell stop
$ adb shell setprop log.redirect-stdio true
$ adb shell start
```

系统会保留这一设置直到模拟器或手机退出，也可以在设备中增加/data/local.prop 以使得这一设备成为默认配置。

Logcat 命令选项列表

选项	描述
-b <buffer>	加载不同的缓冲区日志，例如 event 或 radio。main 缓冲区是默认项，参见 Viewing Alternative Log Buffers.
-c	清空（刷新）所有的日志并且退出
-d	在屏幕上输出日志并退出
-f <filename>	将日志输出到文件<filename>，默认输出是 stdout.
-g	输出日志的大小
-n <count>	设置最大的循环数据<count>，默认是 4，需要-r 选项
-r <kbytes>	每<kbytes>循环日志文件，默认是 16，需要 -f 选项
-s	设置默认的过滤器为无输出
-v <format>	设置输出格式，默认的是 brief，支持的格式列表参见 Controlling Log Output Format.

停止 adb 服务

在某些情况下，可能需要终止然后重启服务端进程，例如 adb 不响应命令的时候，可以通老家重启解决问题。

使用 kill-server 可以终止服务端，然后使用其他的 adb 命令重启。

原文

http://developer.android.com/guide/developing/tools/adb.html