

ADB 使用 For 8810

什么是 ADB

adb 的全称为 Android Debug Bridge，就是起到调试桥的作用。通过 adb 我们可以在 Eclipse 中方便的通过 DDMS 来调试 Android 程序。借助 adb 工具，我们还可以管理手机的状态，还可以进行很多手机相关的操作，如安装软件、系统升级、运行 shell 命令等等。

在 Windows 上安装

1) 软件获取，从 Android 的 SDK 中提取

2) 在手机上开启 USB 调试

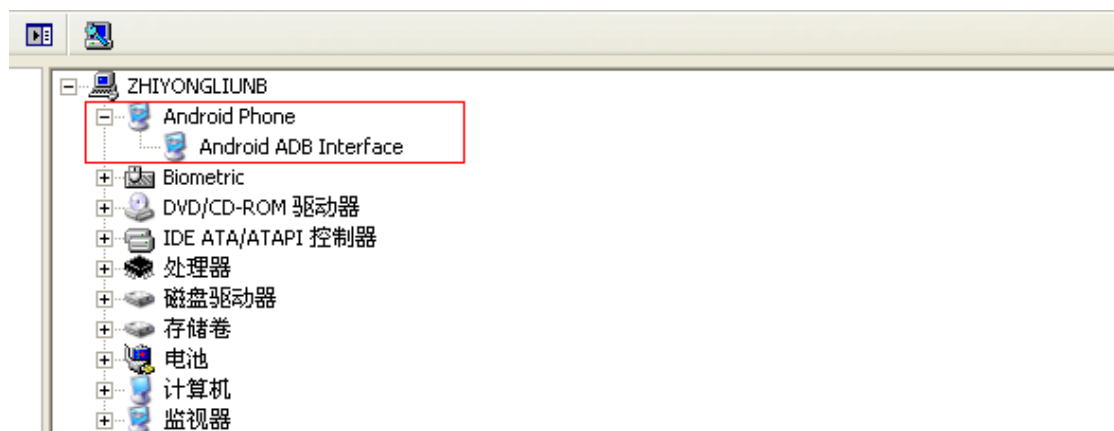
设置->应用程序->开发->USB 调试，选中连接 USB 后启用调试模式

3) 将手机连接到 PC，在 PC 要求安装驱动的时候，安装如下驱动中的 androidwinusb。



SCI-android-usb-driver-jungo.rar

安装成功之后可以从设备管理器看到如下设备



4) 配置 VID

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\alex.liu>cd %USERPROFILE%

C:\Documents and Settings\alex.liu>mkdir .android

C:\Documents and Settings\alex.liu>cd .android

C:\Documents and Settings\alex.liu\.android>
C:\Documents and Settings\alex.liu\.android>
C:\Documents and Settings\alex.liu\.android>echo 0x1782 > adb_usb.ini

C:\Documents and Settings\alex.liu\.android>cd ..

C:\Documents and Settings\alex.liu>
```

上面的%USERPROFILE%在安装了Android SDK的时候换成 %ANDROID_SDK_HOME%

5) 测试连接

```
E:\adb_win_v1.0.25>adb devices
* daemon not running. starting it now *
* daemon started successfully *
List of devices attached
19761202    device

E:\adb_win_v1.0.25>
```

在 Ubuntu 上安装

在 Ubuntu 上不用安装驱动，只需要设置 VID

```
#echo $HOME
#mkdir .android
#echo 0x1782 > .android/adb_usb.ini
#sudo adb devices
#adb shell
```

ADB 命令列表

Category	Command	Description	Comments
Options	-d	发送命令到连接到 USB 的设备，如果 USB 上连接了多个设备返回错误	
	-e	发送命令到模拟器，如果有多个模拟器返回错误	
	-s <serialNumber>	通过设备序列号来标识要连接的设备	如果没有指定号码，则会报错
	-p product name or path	通过设备名来标识要连接的设备	
General	devices	查看所有连接模拟器/设备的设施的清单	查看 Querying for Emulator/Device Instances 获取更多信息
	help	查看 adb 所支持的所有命令	
	version	查看 adb 的版本序列号	
	Connect <host>:<port>	连接到 tcp 地址和端口指定的设备	
Debug	logcat [<option>] [<filter-specs>]	将日志数据输出到屏幕上	
	bugreport	查看 bug 的报告，如 dumphsys ,dumpstate ,和 logcat 信息	
	jdwp	查看指定设施的可用的 JDWP 信息	可以用 <code>forward jdwp:<pid></code> 端口映射信息来连接指定的 JDWP 进程，例如： <code>adb forward tcp:8000 jdwp:472</code> <code>jdb -attach localhost:8000</code>
Data	install <path-to-apk>	安装 Android 为（可以模拟器/设施的数据文件.apk 指定完整的路	

	pull <remote> <local>	将指定的文件从模拟器/设施拷贝到计算机上	
	push <local> <remote>	将指定的文件从计算机上拷贝到模拟器/设备中	
Ports and Networking	forward <local> <remote>	用本地指定的端口通过 Socket 方法远程连接模拟器/设施	端口需要描述下列信息: tcp:<portnum> local:<UNIX domain socket name> dev:<character device name>
	ppp <tty> [parm]...	通过 USB 运行 ppp: <tty> — the tty for PPP stream. For example dev:/dev/omap_csmi_tty1. [parm]... — zero or more PPP/PPPD options, such as default route, local, notty, etc. 需要提醒的不能自动启动 PDP 连接	
Scripting	get-serialno	查看 adb 实例的序列号	查看 Querying for Emulator/Device Instances 可以获得更多信息
	get-state	查看模拟器/设施的当前状态	
	wait-for-device	如果设备不联机就不让执行,也就是实例状态是 device 时	可以提前把命令转载在 adb 的命令器中,在命令器中的命令在模拟器/设备连接之前是不会执行其他命令的, 示例如下: adb wait-for-device shell getprop 需要提醒的是, 这些命令在所有的系统启动起来之前是不会启动 adb 的, 所以在所有的系

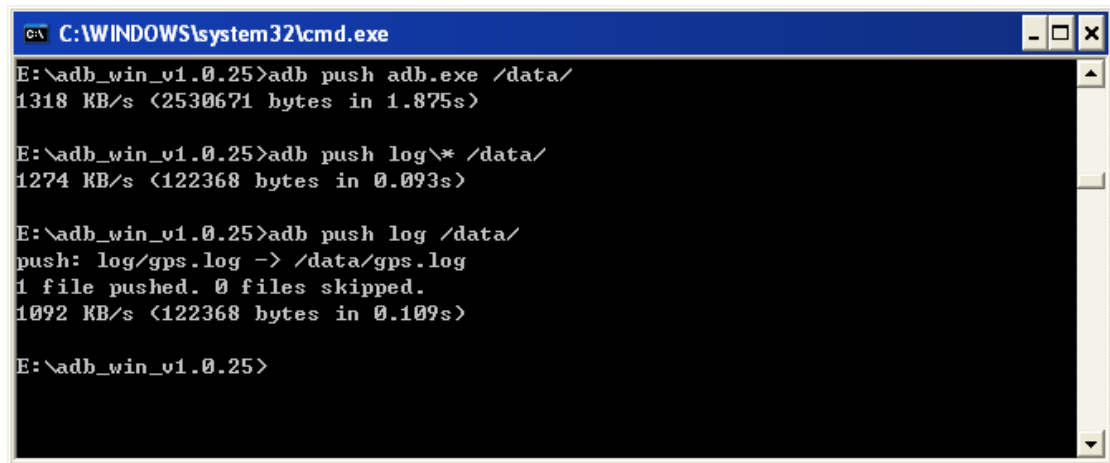
			<p>统启动起来之前也不能执行其他的命令, 例如, 运用 <code>install</code> 的时候就需要 <code>Android</code> 包, 这些包需要系统完全启动, 例如:</p> <pre>adb wait-for-device install <app>.apk</pre> <p>上面的命令只有连接上了模拟器/设备连接上了 <code>adb</code> 服务才会被执行, 而在 <code>Android</code> 系统完全启动前执行就会有错误发生</p>
	<code>start-server</code>	选择服务是否启动 <code>adb</code> 服务进程	
	<code>kill-server</code>	终止 <code>adb</code> 服务进程	
	<code>root</code>	以 <code>root</code> 权限模式运行	
	<code>usb</code>	在 <code>USB</code> 上起 <code>server</code>	
	<code>tcp <port></code>		
	<code>reboot</code>	重启 <code>devices</code>	
Shell	<code>shell</code>	通过远程 <code>Shell</code> 命令来控制模拟器/设备实例	查看获取更多信息 for more information
	<code>shell</code> [<shellCommand>]	连接模拟器/设施执行 <code>Shell</code> 命令, 执行完毕后退出现远程 <code>Shell</code> 端	

应用以及命令详解

传送文件到手机

adb push <local file> <remote dir>

adb push <local dir> <remote dir>



```
C:\WINDOWS\system32\cmd.exe
E:\adb_win_v1.0.25>adb push adb.exe /data/
1318 KB/s (2530671 bytes in 1.875s)

E:\adb_win_v1.0.25>adb push log\* /data/
1274 KB/s (122368 bytes in 0.093s)

E:\adb_win_v1.0.25>adb push log /data/
push: log/gps.log -> /data/gps.log
1 file pushed. 0 files skipped.
1092 KB/s (122368 bytes in 0.109s)

E:\adb_win_v1.0.25>
```

从手机下载文件到 PC

adb pull <remote file> <local dir>

adb pull <remote dir> <local dir>

```
C:\WINDOWS\system32\cmd.exe
E:\adb_win_v1.0.25>adb pull /data/log .\
pull: building file list...
pull: /data/log/gps.log -> .\gps.log
1 file pulled. 0 files skipped.
283 KB/s (122368 bytes in 0.421s)

E:\adb_win_v1.0.25>adb pull /data/log .\log
pull: building file list...
pull: /data/log/gps.log -> .\log/gps.log
1 file pulled. 0 files skipped.
1274 KB/s (122368 bytes in 0.093s)

E:\adb_win_v1.0.25>adb pull /data/*.log .\log
remote object '/data/*.log' does not exist

E:\adb_win_v1.0.25>adb pull /data/gps.log .\log
remote object '/data/gps.log' does not exist

E:\adb_win_v1.0.25>adb pull /data/log/*.log .\log
remote object '/data/log/*.log' does not exist

E:\adb_win_v1.0.25>adb pull /data/log/gps.log .\log
1529 KB/s (122368 bytes in 0.078s)

E:\adb_win_v1.0.25>
```

安装应用程序

adb install <apk file>

```
C:\WINDOWS\system32\cmd.exe
E:\adb_win_v1.0.25>adb install android_10503_20120203.apk
1245 KB/s (1693407 bytes in 1.328s)
  pkg: /data/local/tmp/android_10503_20120203.apk
Success

E:\adb_win_v1.0.25>
```

卸载应用程序

adb uninstall <package>

adb uninstall -k <package> 删除时保留 data 和 cache 里的数据

```
C:\WINDOWS\system32\cmd.exe
E:\adb_win_v1.0.25>
E:\adb_win_v1.0.25>
E:\adb_win_v1.0.25>
E:\adb_win_v1.0.25>adb uninstall zausan.zdevicetest
Success

E:\adb_win_v1.0.25>adb install android_10503_20120203.apk
1245 KB/s (1693407 bytes in 1.328s)
    pkg: /data/local/tmp/android_10503_20120203.apk
Success

E:\adb_win_v1.0.25>adb uninstall -k zausan.zdevicetest
The -k option uninstalls the application while retaining the data/cache.
At the moment, there is no way to remove the remaining data.
You will have to reinstall the application with the same signature, and fully un
install it.
If you truly wish to continue, execute 'adb shell pm uninstall -k zausan.zdevice
test'

E:\adb_win_v1.0.25>
```

登录到手机的 shell

adb shell

```
C:\WINDOWS\system32\cmd.exe
6 个文件      4,504,402 字节
3 个目录 127,515,099,136 可用字节

E:\adb_win_v1.0.25>adb shell
# ls /system/
ls /system/
media
opl
lib
busybox
bin
framework
fonts
usr
vendor
xbin
sps
build.prop
app
lost+found
etc
# exit
exit

E:\adb_win_v1.0.25>
```

直接执行手机上的命令，把输出输出到 PC

adb shell <cmd>


```
C:\WINDOWS\system32\cmd.exe

E:\adb_win_v1.0.25>adb shell ls /system/
media
opl
lib
busybox
bin
framework
fonts
usr
sps
xbin
build.prop
app
lost+found
etc

E:\adb_win_v1.0.25>
```

```
C:\WINDOWS\system32\cmd.exe

E:\adb_win_v1.0.25>adb shell /system/xbin/busybox ls -l /system/
total 1570
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34mapp+[0m
drwxr-xr-x  1 0      2000      2048 Jan  1  08:00 +[1;34mbin+[0m
-rw-r--r--  1 0      0          1915 Dec  9  2011 +[0;0mbuild.prop+[0m
-rw-r--r--  1 0      0      1580532 Nov  2  2011 +[0;0mbusybox+[0m
drwxr-xr-x  1 0      0          2048 Feb 12  08:45 +[1;34metc+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34mfonts+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34mframework+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34mlib+[0m
drw-rw-rw-  1 0      0          2048 Feb 12  08:44 +[1;34mlost+found+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34media+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34mopl+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34msps+[0m
drwxr-xr-x  1 0      0          2048 Dec  9  2011 +[1;34musr+[0m
drwxr-xr-x  1 0      2000      2048 Dec  9  2011 +[1;34mxbin+[0m

E:\adb_win_v1.0.25>
```

抓 Log

抓 radio 的 Log

```
adb logcat -b radio -v time
```

抓其他应用的 Log

```
adb logcat -v time
```

抓特定模块的 Log

```
adb logcat -s -v time mod1 mod2 ...
```

抓 Kernel 的 Log

```
adb shell dmesg
```

```
adb shell cat /proc/msg
```

Log 重新定向

```
adb logcat |tee filename    (Linux 下)
```

```
adb logcat > filename    (windows and linux 下)
```

其他命令

```
adb kill-server
```

```
adb start-server
```

```
adb reboot = adb shell reboot
```