RTD: USAGE

CONTENTS

De Jesus, Ramon L
ramon.dejesus@nist.gov

RTD

WHAT IS "RTD"?

"RTD" is a service app launcher that automates the repeated launching of Android test apps. Currently, it works only with apps that implement a particular finish/exit (System.exit(0)) protocol (future developments may remove this limitation). It was developed and tested on Android version 2.2 using a Motorola DROID X.

Included with RTD are compatible standalone versions of the test cases found in the Android spin of the validation test suite for application profiling tools that is available from the Software Performance Project's web page. Spins of the test suite are also available for Linux [x86_64 Linux with the GCC compiler] and Windows [x86_64 Windows with the Intel toolset (Composer XE 2013, VTune Amplifier XE 2013)].

WHAT DOES IT DO?

Unlike the original Android suite, RTD launches each individual test in its own thread, allowing more precise tracing of the code of interest. Using a facility such as Android's Debug class, the observed results from function-level profiling of self time, total time, and call graphs can be compared with the expected results to gain confidence that the profiler is performing as expected. If the results are surprising, different parameter settings can be used to narrow down the root cause of the behavior.

For additional background on the functionality and motivation of the validation test suite, please see the NIST Technical Note Configuration of profiling tools for C/C++ applications under 64-bit Linux.

De Jesus, Ramon L
ramon.dejesus@nist.gov

USING "RTD"

TOOLS NEEDED

HARDWARE

- COMPUTER WITH LINUX (INSTALLED OR VM)
- ANDROID PHONE (TO GET PHONE ACTUAL MEASUREMENTS)

SOFTWARE

- JAVA SE DEVELOPMENT KIT 7 + (JDK) [HOW TO]
- ANDROID SDK (ECLIPSE + ADT + ADB) [HOW TO]

OPTIONAL

- IA32-LIBS (IF LINUX OS 64-BIT) //SOFTWARE [HOW TO]
- GRAPHVIZ (TO OUTPUT GRAPHS WITH DMTRACEDUMP) //SOFTWARE [HOW TO]
- GETTRACEFILES (IF USER WANTS AUTOMATIC FETCH AND CONVERSION OF TRACEFILES) //SOFTWARE [HOW TO]

INSTALLING "RTD"

INSTALLING FROM SOURCE

Because we released the source code of "RTD", you can easily download, modify and compile the .apk and install it on your device. In this report, that process is not going to be explained.

For a tutorial on building apps from source, please visit:
http://developer.android.com/training/basics/firstapp/index.html

INSTALLING THE .APK

1. A prebuilt .apk file is also included for easier installation. To install the .apk, you can download it and install it directly from your phone or via adb commands. To install it via adb commands, connect your phone to your computer and ensure that your device is recognized by typing “adb devices” on a Linux terminal:

```
$ adb devices
```

If your device is connected, it should appear under “List of devices attached”.

Note: This and other “adb” commands will only work if you did a perfect “Computer Setup” by following the set of instructions found inside this manual. (See “Computer Setup” to verify)

2. Once you know that your device is ready, proceed to type “adb install /path/to/app.apk” on a Linux terminal:

```
$ adb install /path/to/app.apk
```

If all was run correctly, your application should be ready to run and play on your device’s app list.

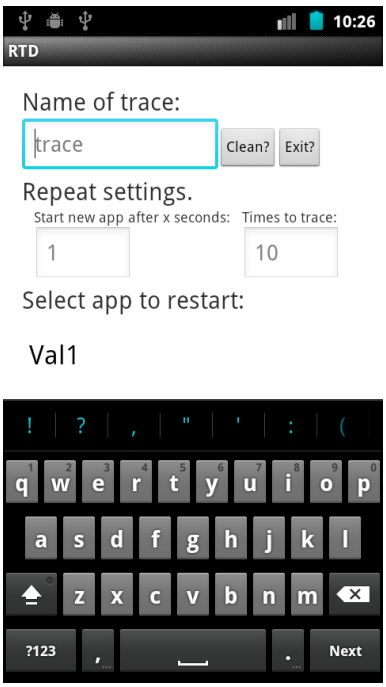
RUNNING A TEST CASE

To run a test case in “RTD”, you only need to fill in the blanks and hit start.

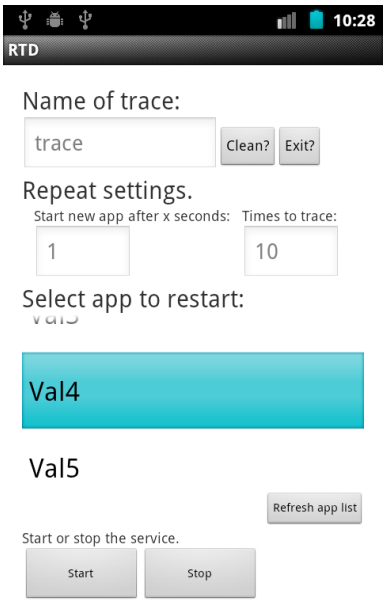
EXAMPLE WITH VAL4

1. Start the application.

2. Indicate the name of your trace file. (Remember that trace data will be sent to /sdcard/Traces.)

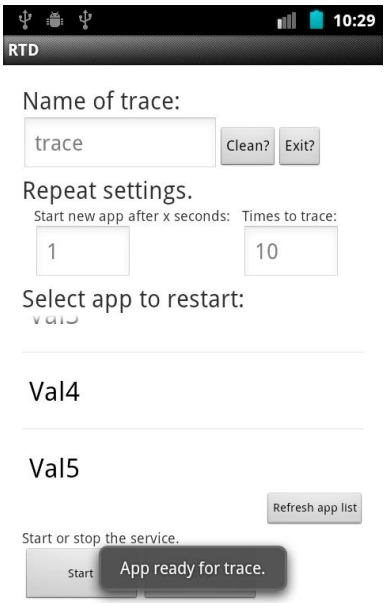


3. Scroll down and select Val4 app from the apps list.

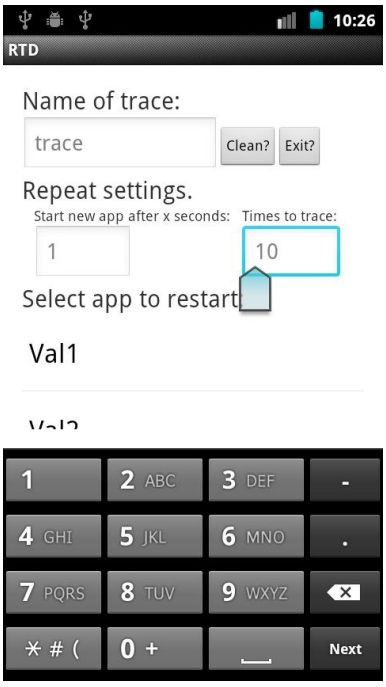


If you are unable to see any apps to restart please make sure that you first installed those that are included with the source inside the folder "Test-Case-Apps".

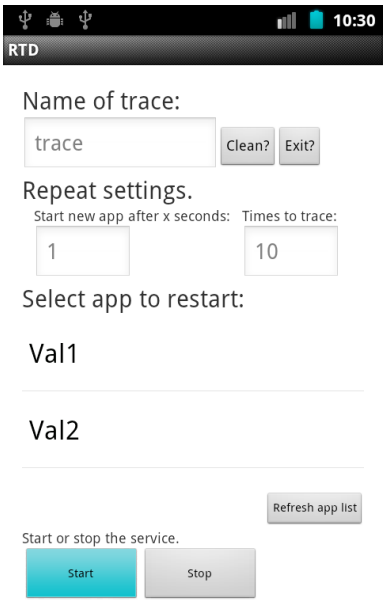
4. A confirmation message and two vibrations will let you know that the service is ready to trace the app.



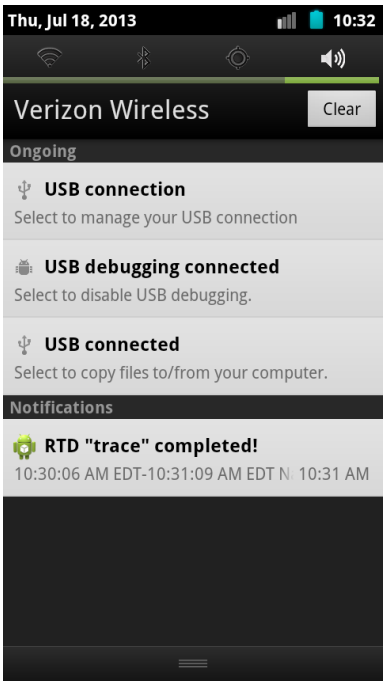
5. Indicate the number of times to repeat the app. You can also configure the sleeping time between app launches in “Start new app after x seconds”.



6. Hit Start!



7. When it's done, a message box will appear and a toast notification will inform you of the time it took to complete.



8. Done. The trace files named after the test case you ran are on your device's sdcard, in /sdcard/Traces/.

FETCHING THE TRACEFILE

USING THE CUSTOM LINUX SCRIPT "GETTRACEFILES"

The script "getTraceFiles" can be found under the util directory.

"getTraceFiles" will permit the user to automatically gather and convert individual trace files or a batch set of trace files stored in the /sdcard/Traces/ directory of our Android device. (Notes and warnings are embedded in the script.)

Note: This script will only work if you did a perfect "Computer Setup" by following the set of instructions found inside this manual. (See "Common Errors" to check for common errors.)

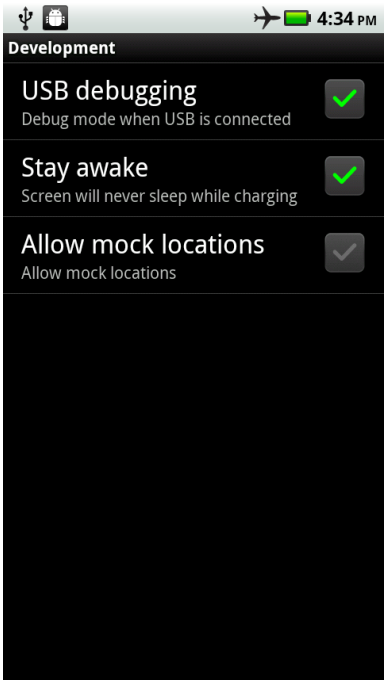
FETCHING THE TRACEFILES WITH GETTRACEFILES

1. Connect your phone to your computer.
2. Go to USB connection.

De Jesus, Ramon L
ramon.dejesus@nist.gov

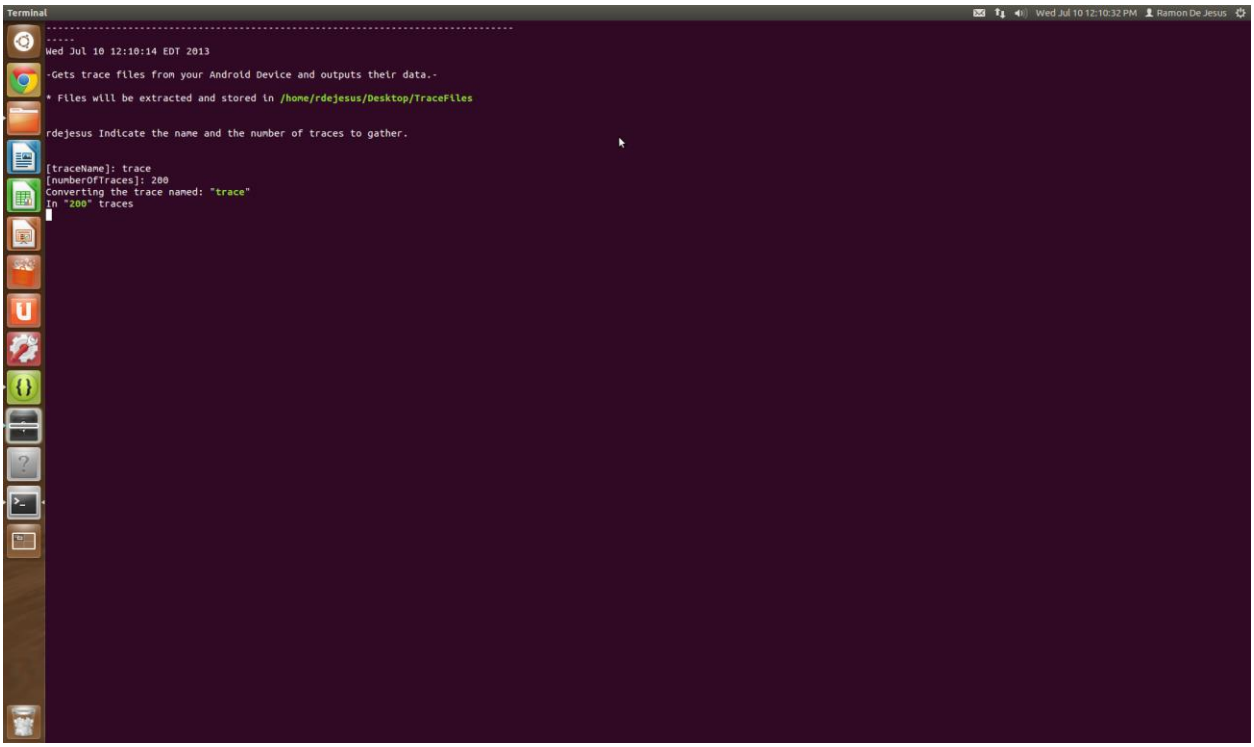
3. Select Charge Only.

4. Ensure that USB debugging is on, that you have a proper installation of adb and dmtracedump in your PATH variable, and that your phone is being recognized by the terminal command “adb devices”. (See more on “Path for Computer Setup”.)



De Jesus, Ramon L
ramon.dejesus@nist.gov

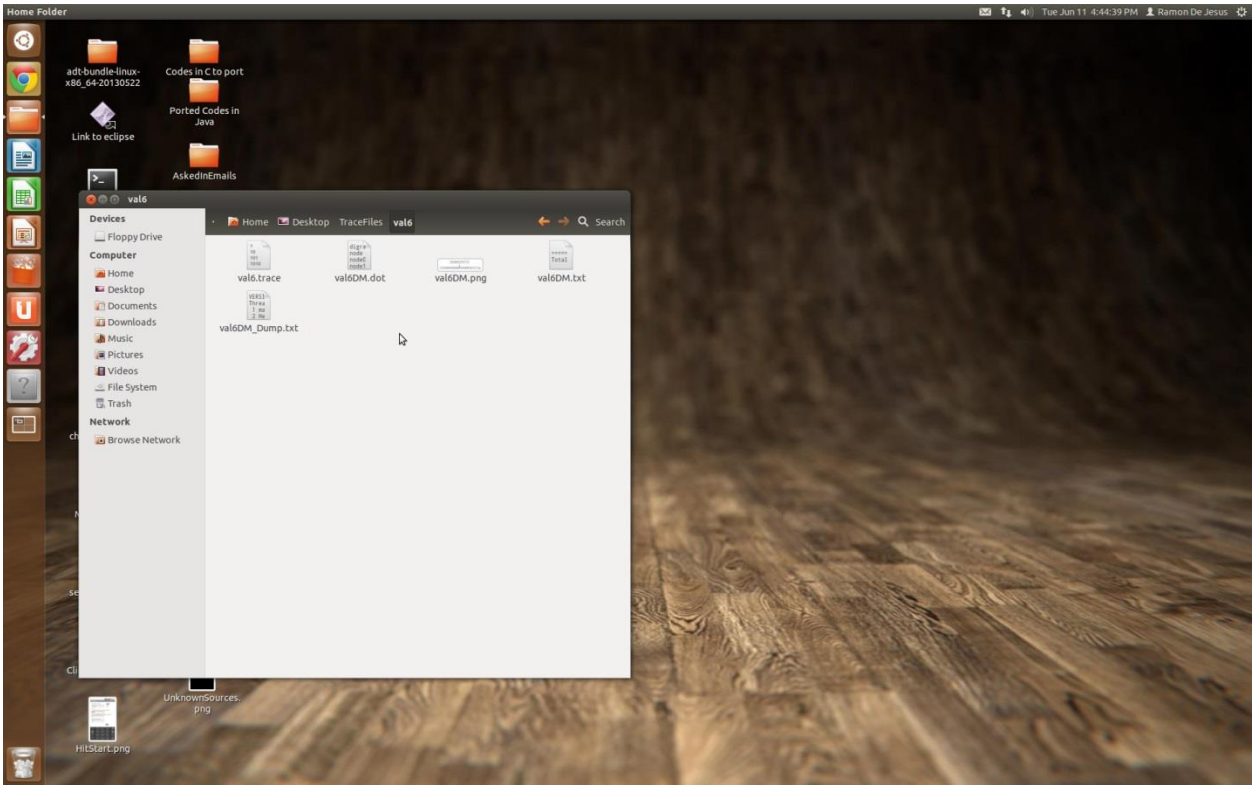
5. Open a terminal, move to the script directory using cd command and run getTraceFiles. Just fill in the prompts and enter.



Basic usage of “getTraceFiles” script could also be done on a terminal by typing:
getTraceFiles [traceName] [numberOfTraces]

```
$ getTraceFiles [traceName] [numberOfTraces]
```

6. If everything was good, a folder named TraceFiles with all of your trace files and their outputs should be on your desktop.



PLOTTING THE TRACE

USING THE CUSTOM LINUX SCRIPT "PLOTDATA"

The script "plotData" is available in the util directory.

"plotData," an R script caller, will permit the user to automatically convert individual .csv files stored in the ~/Desktop/TraceFiles/[traceName-DATA] directory of the PC to graphical plots. (Notes and warnings are embedded in the script.)

Note: This script will only work if you did a perfect "Computer Setup" by following the set of instructions found inside this manual and the folder [traceName-DATA] exists in ~/Desktop/TraceFiles/ directory. (See "Common Errors" to check for common errors.)

All of the R scripts that plotData uses can be found inside the extra folder.

Good to know about the plots:

- *Scatter plots and density plots omit the top 1% of data while the histogram shows the entire range of the trace.*

PLOTTING THE TRACEFILES WITH GETTRACEFILES & PLOTDATA

1. If you were using “getTraceFiles,” the process to convert those “.csv” files to plots should be relatively easy: just fill in the prompts in the terminal and you are done.
2. Else, if you stop using “getTraceFiles” and you have your “.csv” files inside the ~/Desktop/TraceFiles/[traceName-DATA] directory of your PC, simply double click on the script “plotData” inside of the “util” directory and fill in the prompts.

ERASING OLD TRACEFILES

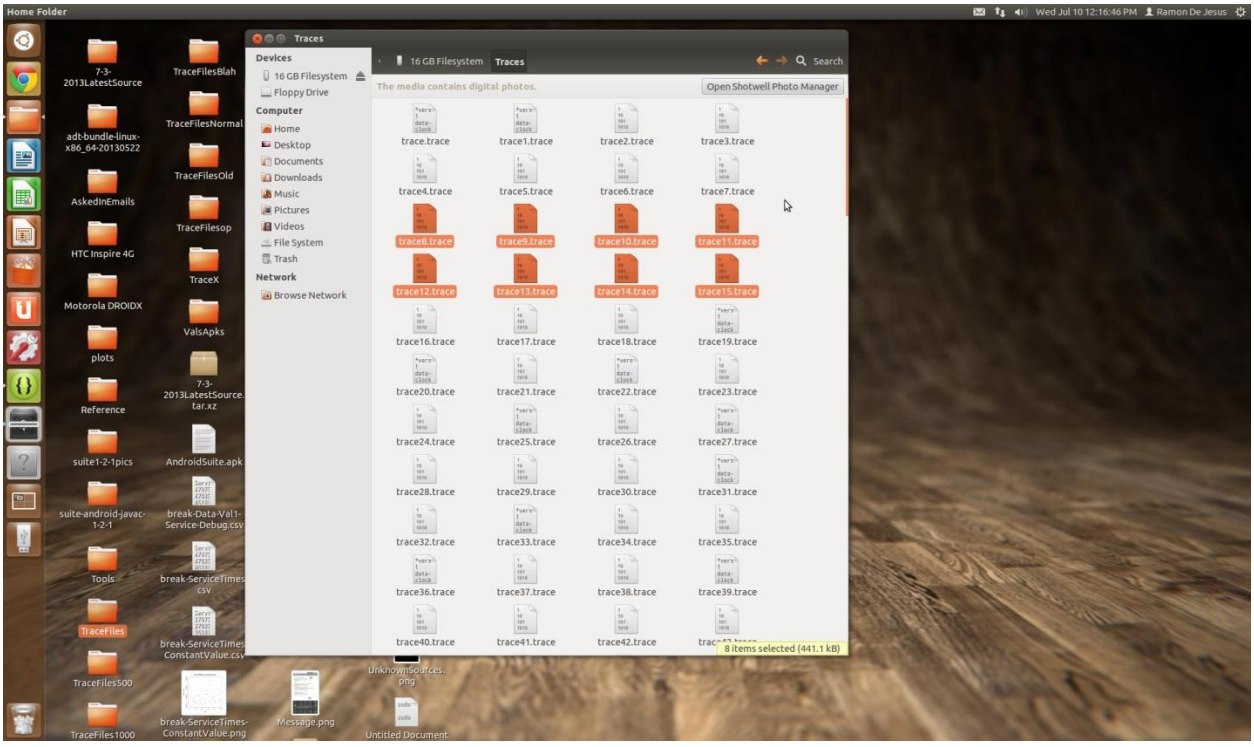
ERASING THE FILES WITH YOUR COMPUTER

1. Connect your phone to your computer.
2. Go to USB connection.

De Jesus, Ramon L
ramon.dejesus@nist.gov

3. Select USB Mass Storage and hit OK.

4. Once in the computer it should pop up like a normal USB drive. Select the traces you want to delete, inside the /sdcard/Traces folder.



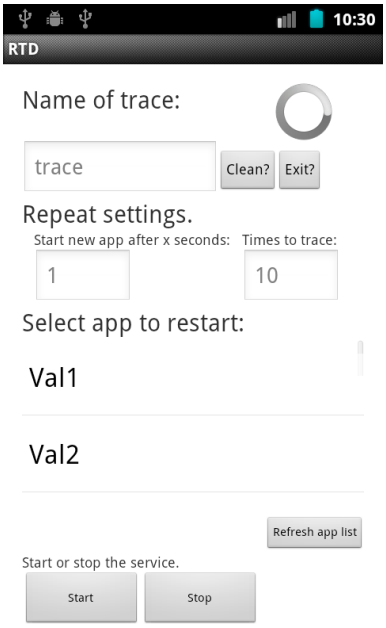
5. Delete them and you're done.

De Jesus, Ramon L
ramon.dejesus@nist.gov

ERASING THE FILES WITH "RTD"

1. Start the application.

2. Hit the "Clean?" button. This will delete the trace files automatically.



3. Done!

MISC

COMPUTER SETUP

This project assumes you are running on a Linux working environment.

Note: For this work, we use Ubuntu 12.04 LTS (http://www.ubuntu.com/download/desktop), mounted on VM Player (http://www.vmware.com/go/downloadplayer/), a virtual machine.
Other distributions of Linux could be used; please have in mind your processor type (32bit | 64bit).
More distributions: (http://lmgtfy.com/?q=Linux+distributions#)

1. Download and install the correct "JAVA SE DEVELOPMENT KIT 7" for your OS (http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html?ssSourceSiteId=otnes)
2. *"If running a 32-bit OS, skip this step."*
   As root install ia32-libs via the terminal.

```
$ sudo apt-get install ia32-libs
```

3. Download and extract Android SDK. Remember where you extracted it and don't change or delete any folders. (http://developer.android.com/sdk/index.html)
4. *"If you want to output graphics with dmtracedump"*
   Download and install graphviz. Take care of your distribution.
   http://www.graphviz.org/Download..php
5. *"For easy later access of our terminal commands, adb and dmtracedump…"*
   As root modify or create your "environment" PATH variable located on `/etc/environment`.
   from PATH=""
   to PATH="<EXTRACTED_LOCATION>/adt-bundle-linux-x86_64-20130522/sdk/platform-tools:<EXTRACTED_LOCATION> /adt-bundle-linux-x86_64-20130522/sdk/tools"

6. *"If you want to use a phone for the profiling"*
   As root modify or create your "51-android.rules" file located at `/etc/udev/rules.d/51-android.rules.` Add one line like the following for each phone that you have:
   `SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"`
   Notice `"0bb4"`, because that's where your device vendor ID should be and it's the only string we should be changing according to our phone.

This table provides a reference to the vendor IDs needed in order to add USB device support on Linux. The USB Vendor ID is the value given to the ATTR{idVendor} property in the rules file, as described above.

Company	USB Vendor ID
Acer	0502
ASUS	0b05
Dell	413c
Foxconn	0489
Fujitsu	04c5

*More devices on the next page.

De Jesus, Ramon L
ramon.dejesus@nist.gov

Fujitsu Toshiba	04c5	Nvidia	0955
Garmin-Asus	091e	OTGV	2257
Google	18d1	Pantech	10a9
Haier	201E	Pegatron	1d4d
Hisense	109b	Philips	0471
HTC	0bb4	PMC-Sierra	04da
Huawei	12d1	Qualcomm	05c6
K-Touch	24e3	SK Telesys	1f53
KT Tech	2116	Samsung	04e8
Kyocera	0482	Sharp	04dd
Lenovo	17ef	Sony	054c
LG	1004	Sony Ericsson	0fce
Motorola	22b8	Teleepoch	2340
MTK	0e8d	Toshiba	0930
NEC	0409	ZTE	19d2
Nook	2080		

- Vendor ID's provided by "Android Developer"

7. Reboot your machine and the computer set up is completed.

COMMON ERRORS

"ADB" COMMANDS

1. COMMAND NOT FOUND

- *Verify that you have correctly installed the Android SDK (Eclipse + ADT + ADB)*
  [Android SDK]
- *If you installed the Android SDK correctly, then your PATH environment variable may be set incorrectly.*
  [Setting the PATH]

2. DEVICE NOT FOUND

- *Please ensure that a clear connection between your device and your machine has been set. Did you plug in the USB cable?*
- If your device is well connected to your computer and *USB debugging mode is ON*, then your problem has to be with a non-valid signature of the drivers.
  [*Phone Drivers*]

"GETTRACEFILES" COMMANDS

1. COMMAND NOT FOUND

- *For direct use of getTraceFiles in a terminal, you either have to navigate to the file via "cd" and run getTraceFiles's commands, set the file location in your Path variable, or double click and select "Run in terminal".*

2. DEVICE NOT FOUND

- *"getTraceFiles" works by using adb and other Android platform tools commands for its output data. Please verify that your adb installation is in a working state and that all Paths have been properly set up in your environment file.*
  [ADB Errors]

3. TRACE NOT FOUND

- Please ensure that you type the exact name of the trace file you want to gather. In addition, verify that it exists on your phone.

"PLOTDATA" COMMANDS

1. NO DATA FOLDERS FOUND

- *Make sure that a folder named [nameOfTrace-DATA] is inside ~/Desktop/TraceFiles/*
- *Make sure you typed in the traceName of your trace correctly.*

REFERENCES

ANDROID SPIN

http://www.nist.gov/itl/ssd/cs/upload/suite-android-javac-1-2-1-1.txz

LINUX SPIN

http://www.nist.gov/itl/ssd/cs/upload/suite-linux-gcc-1-2.tgz

WINDOWS SPIN

http://www.nist.gov/itl/ssd/cs/upload/suite-windows-icl-1-2.zip

ANDROID DEVELOPER

http://developer.android.com/tools/device.html

De Jesus, Ramon L
ramon.dejesus@nist.gov

FURTHER READING

David Flater, "Configuration of profiling tools for C/C++ applications under 64-bit Linux," NIST TN 1790, National Institute of Standards and Technology, Gaithersburg, MD, March 2013. http://dx.doi.org/10.6028/NIST.TN.1790

(Software Performance Project) http://www.nist.gov/itl/ssd/cs/software-performance.cfm

(Android Programming) http://developer.android.com/guide/components/index.html

(Profiling with Traceview and dmtracedump) http://developer.android.com/tools/debugging/debugging-tracing.html

(Installing ia32-libs) http://forums.kali.org/showthread.php?827-How-to-install-ia32-libs

(Core DDMS usage) http://developer.android.com/tools/debugging/ddms.html

National Institute of Standards and Technology
U.S. Department of Commerce

De Jesus, Ramon L
ramon.dejesus@nist.gov