

Implementation of Android SDK into Debian Linux

Adnan Hodzic

State University of New York at Canton

CITA 481

May, 2012

Abstract

Android is the biggest and fastest growing mobile operating system and on top of it all it's based on Linux. But even besides this fact support for devices running it as well as its development on Linux, Android's "home" platform is far from ideal. It's important to note that Linux is fully and officially supported by Google (its parent company) and Open Handset Alliance providing all the necessary tools and necessary documentation.

However there's a gap in this whole process, gap which constrains Android users or/and developers to complete this whole process in a error free and semi-automated manner by selecting which component they would you like to have installed and fully working with a single hit of a button. There are legions of Android users/developers who are experts in Android systems without much or any expertise in Linux, individuals who are using Linux strictly because it makes their business or home use much less expensive, more secure, reliable and other factors which Linux provides. With more and more of Linux distributions becoming oriented to new users to even a degree of this factor becoming a trend, it's important that these individuals have tools which will enable them to work on things they are experts at in out of the box solutions without fiddling with any of Linux internals or having a need to configuring anything.

Thus, the goal of this project is to make an installer which will seamlessly implement and integrate Android SDK as well as its components without its end user having to do anything but install the package and give their answers when prompted. Installer isn't designed to be strictly developer oriented and should provide some features from which even a regular user can benefit from.

At the time of writing this paper I successfully completed first version of an installer "android-sdk-linux" version 0.1 (Appendix: android-sdk-installer)

Installer itself consists of few parts which are:

- Installation and configuration (implementation and integration) of Android SDK (Software Development Kit) into Debian Linux
- Installation and configuration of ADT (Android Developers Tool) plug-in for Eclipse IDE
- Ability to install hardware device support for all Android devices
- Extra features such as adding support for MTP (Media Transfer Protocol)

Table of Contents

Abstract2

Table of Contents.....4

Chapter 1.....5

What is Android?.....5

What is Debian Linux?7

Explain difference between Android and Debian 10

Android gets merged with Linux kernel main tree 11

Android SDK..... 12

Chapter 2.....14

Eclipse ADT plug-in installation 18

Explain how this process differs on Windows, Mac OS X and Linux.....21

How the idea of an “android-sdk-installer” was born?24

What kind of advantage does this bring to Linux?25

Application Design and components function.....26

Chapter 3.....39

Which programming languages were used in creation of this application?39

How was this platform chosen from technical stand point?39

Standard application installation procedure on Linux42

Application creation and installation in Debian44

Application implementation into Debian repository sources46

Conclusion.....48

List of References50

Appendix: android-sdk-installer53

Chapter 1

What is Android?

Android is Linux based operating system with primary focus on mobile devices such as smartphones and tablets, even though Android has proven its uses in many other devices such as TV's (Google TV). It is also speculated that this focus might change to netbook and notebook market with upcoming version of Android v5.0 codename: "Jelly Bean".

Android Inc. was founded in October 2003 in Palo Alto, California with Andy Rubin in the lead along with Chris White, Rich Miner and Nick Sears with most of them coming from companies related to communications and telephony. Their main mission was to develop a "smarter" mobile device, along with mostly keeping their work under seal of secrecy, except the fact that they were working on developing software for mobile devices. Later on in August, 2005 Android Inc. was acquired by Google Inc, becoming one of its subsidiaries, again with most of its work remaining under the veil of secrecy even though many speculated that Google was preparing to enter mobile phone market.

Beginning of November 2007, Open Handset Alliance was formed; a consortium which consisted of several big companies such as Google, Intel, Samsung, HTC, Motorola, Nvidia and T-Mobile just to name a few. Couple of days later Android 1.0 beta was released. It was a new mobile operating system developed by Google and Open Handset Alliance, which wasn't released for any particular phone and it was mainly embraced and further researched by the developer community. It was released under Open Source license (Apache Software License, 2.0) which is lead by "Android Open Source Project (AOSP)" which is assigned for its development and maintenance. For the further reference, it may be worth nothing that first version of Apple iPhone along with first version of iOS was released in the

summer of 2007.

Following year Open Handset Alliance was joined by another dozen of leading companies in technology and mobile communication fields, today consortium numbers 84 companies. And it wasn't until this year (2008) that Android v1.0 was released, which was its truly first milestone and which was released on Android's first device "HTC Dream (G1)", same was done with Android v1.1 which was released in beginning of 2009 and was initially released for a single device (T-Mobile G1).

However it wasn't until beginning of April in 2009, that Android started giving their new releases codenames based on dessert names following alphabetical order, it wasn't until Android 1.5 "Cupcake" that Android started spreading to number of devices and manufacturers and thus allowing Android to slowly enter mainstream when it comes to market share.

To give you a solid comparison, strategy of open standards and open source gave Google an immense comparative advantage in terms to its main competitor Apple, unlike iPhone which was tailored for a single device and single maker with closed source operating system. Android was embraced by number of leading mobile device makers, allowing them to easily customize their own version of the operating system their device is running on. Thus in the end allowing Android to appear in many shapes and forms.

In very short period of time, Android became one of the fastest growing operating systems for mobile devices, according to Google's Senior Vice President of Mobile Andy Rubin to this date there are more than 300 million activated devices with around 700.000 devices being activated each day in 137 countries and regions. It has more than 500.000 applications being available for the same platform with over 10 billion downloads.

These numbers lead to very enviable numbers when it came to market share as well,

according to latest information it is believed that Android has topped 50% market share, with iPhone lagging behind with 30.2%, RIM's Blackberry with 13.4% leaving Microsoft's Windows Phone with only 3.9%.

What is Debian Linux?

Debian Linux is one of the oldest (18+ years) and most influential Linux distributions with more than 120 active derivatives, one of them being Ubuntu which is currently by far the most popular Linux distribution. It has more than 30.000 official software packages spreading over 11 computer architectures.

Debian was born on August 16th 1993 and was announced by its creator Ian Murdock; name was made out his (then) girlfriend Debra and his first name Ian: Deb + Ian. Followed by distributions release Ian also released the Debian Manifesto which outline the new operating system as well as his intentions along with the fact that it was of an open manner released in spirit of GNU/Linux, thus to this date you can still see Debian name being written in “Debian GNU/Linux” form. Before release of Debian only Linux distribution which consisted of compiled various software packages was SLS (Softlanding Linux System) which was poorly maintained and had numerous bugs. Distribution that was later on born from SLS is also one of the oldest active Linux distributions today, Slackware. Due to the way it was announced and in accordance to its manifesto soon after it was released Debian was picked up by the community and many other developers and it is believed Debian is the first community based Linux distribution.

It is well known for its strict adherence to philosophy of Unix and Free Software, besides Linux kernel there is a port to FreeBSD kernel with standard set of Debian packages named “Debian GNU/kFreeBSD”. Debian is available in more than 65 languages and also

consists of more than 10 available desktop environments to choose from with GNOME being the default one.

Another interesting thing about Debian is that it's made by more than a thousands of volunteers and is being supported by couple of nonprofit organizations, one of the most notable ones being the SPI (Software in Public Interest) thus it may be openly said Debian organization is completely decentralized in its organizational structure and all the work is done “online” (remotely) by developers coming from all parts of the world.

Once a year, developers meet in person on a conference called “DebConf” to discuss how to resolve current issues and to discuss their plans for future Debian releases, these conferences are sponsored by one of the biggest and leading companies in world today such as Google, HP, Intel and et cetera, it's also important to note that a lot of Debian developers work in those same companies. Author of this text was an event organizer, local team founder and leader of last DebConf11 which after locations such as New York City, Edinburgh, Helsinki, Toronto and others, conference was held in Banja Luka, Bosnia and Herzegovina.

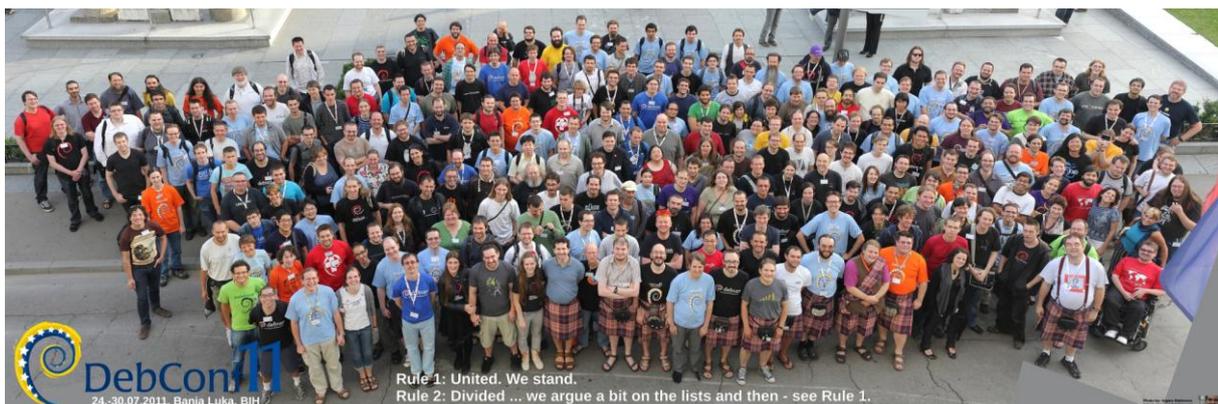


Figure 1: DebConf11 Group photo

Interesting details regarding Debian is also cost of its development, since it's all done by volunteers it can be and is labeled as free but if we use a COCOMO model to calculate its development as of February 2012 production costs for Debian Wheezy (current version it's

being worked on) it would cost ~19 billion US dollars, with each upstream source code author worth an average of ~1.1 million US dollars. Interesting detail because the production cost of 19\$ billion is given to the community by the price tag of “free”.

Its development procedure model is a very unique one and is widespread within other projects as well. Where project distribution can be found in couple of repositories, such as: “*experimental*”, “*unstable*”, “*testing*” and “*stable*”. But is generally divided into three main branches: *stable*, *testing* and *unstable*.

With *experimental* and *unstable* usually being almost the exact versions released by their upstream authors. However *experimental* is being periodically used and is mostly used for purely experimental purposes. While *unstable* codename “*Sid*” is mainly used by Debian developers and users who want the latest “bleeding edge” software, needless to say this software is prone to bugs and problems. Although many argue that *Unstable* doesn't deserve its name as even in this state it's known to be more stable than most of the other “stable” distribution releases out there. After software has been thoroughly tested or none of the bugs have been detected within 2 weeks it's automatically pushed into *Testing* which as the name says is mostly used for testing purposes and even though it's labeled as testing this is stable software and it's important to note that *Testing* is what's about to become next *Stable* release. *Stable* is mostly used on servers and business environment, and more conservative users who prefer stability and security over all other factors.

Unlike many other distributions *Debian Stable* is released every couple of years, in its past this cycle was pretty irregular ranging from one to three years, however with release of Debian 6 codename: *Squeeze* it was announced that *Debian Stable* will be released within two year cycle, this will not only enable developers to better plan their application development and deployment but also enable its users to be able to better plan and prepare for the upgrade

procedure. Even though it's not backed by any parent company *Debian Stable* release is officially supported and will be receiving security updates (Debian's security policy also states this) for one year after the next *stable* release. That means after new *stable* is released, old *stable* becomes “*Old Stable*” and is still supported for at another year giving it approximately a support up to 3+ years.

Debian codenames have been named after “*Toy Story*” movie characters with each of them changing with each *Stable* release, while *Sid (Unstable)* remains to be the same character and perfectly portrays the state of that distribution as in the movie Sid is the mean kid that likes to cause havoc and break all the toys.

There is another experimental distribution within Debian which isn't even officially supported, Debian *CUT (Constantly Usable Testing)*. This is Debian's version of “*Rolling Release*”, concept which has been accepted by couple of new Linux distributions and is being tested by many others. Author of this paper has published an article “[Debian CUT, a new rolling release?](#)” for those who are interested in one such topic.

It is this abundance of features and options that make Debian a perfect platform for devices ranging from airplanes, supercomputers, servers, phones to notebooks and makes a perfect base to make and tailor your own Linux distribution.

Explain difference between Android and Debian

One may say since they are both Linux based platforms/systems theoretically they could be seen as the same but in reality there are vast differences. For example Debian is developed by thousands of volunteers while Android is developed by Open Handset Alliance (which is lead by Google) and Google itself. While Debian supports 11 different computer architectures, currently Android officially supports only one, ARM. There are projects which

are providing x86 support and apparently current Google TV uses x86 architecture, Intel has also announced they will be entering this market with their new sets of chips as they plan to introduce x86 architecture for Android based phones. Projects like *Android-x86* have released unofficial ports of Android to x86 architecture for a limited number of devices.

Android doesn't have X Window System which is used to provide support for graphical user interfaces (GUI) and is used by all Linux distributions that have GUI capabilities. Another major distinction is that Android doesn't support the full set of GNU libraries, again something that's used by most modern Linux distributions. Thus generally put, structurally these two are completely different systems from almost every possible aspect and one thing they definitely have in common in its verbatim form is Linux kernel.

Android gets merged with Linux kernel main tree

Since its inception, Android, even though Linux in its heart meaning it was based on Linux main kernel tree was dropped out of it, due to many disagreements between Linux kernel tree maintainers and Android kernel maintainers. These were mostly based on the facts that Android was developed at a very fast rate and changes that were made to original Linux kernel weren't being pushed back, and by some opinions because Google lacked personnel to address this particular issue. Changes that were made on Android kernel which were pushed back to Linux main tree weren't updated and were left neglected or were simply completely abandoned.

Thus for years, even though at its base same operating systems it could be said these two were developed in separate manner. There were even speculations that Linux Foundation may sue Google due to Gnu General Public License violation, of course these claims and rumors were debunked by *Linus* himself in the fall of 2011 when he stated: "*there's still a lot*

of merger to be done ... eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years.”

However just couple of months after this announcement was made at 2011 Kernel Summit in Prague it was announced that Android kernel tree will be merged into mainline Linux kernel tree, “Android Mainlining Project” was created with aim to help with the merger process.

Of course, both systems will greatly benefit from one such merger but aspects I think could especially benefit from one such merger are Linux distributions. When I say this I'm strictly referring to Android's “*low memory killer*” feature which in my opinion is revolutionary approach to application management. On Android you never quit from applications as you do on all other operating systems, meaning Android applications never exit, instead when you switch to other application or process application you previously used remains saved and stored in memory in an inactive state. Application remain in one such state until device becomes low on space when a special mechanism decides which application to kill in order to free up memory without affecting system where same one is not being used.

When I was first greeted with this approach the first thing that crossed my mind is “image how one such mechanism” would benefit desktop or notebook computers. Of course it's needless to say how one such approach benefits user experience and with upcoming Linux 3.4 release date behind the corner implementation of one such mechanism into Linux distributions is becoming a reality.

Android SDK

Android SDK (Software Development Kit) is a extensive set of development tools which allow you to develop and publish applications for the same platform. Some of these

include AVG Manager (android), Android Debug Bridge (adb), Dalvik Debug Monitor (ddms), Android Emulator (emulator), various libraries along with diverse selection of tutorials, documentation and sample code. Besides Linux other platforms that are supported are Mac OS X and Windows.

Also worth mentioning is Android Development Tools (ADT) Eclipse IDE plug-in; with Eclipse IDE being one of the most popular and used IDE's (Integrated Development Environment) for Java development. ADT plug-in extends Eclipse IDE capabilities by allowing developers to quickly create new Android projects, as well as create/modify/build/debug their applications in Java and XML files rather than rely on using command line tools to achieve the same purpose. Among many other things it also provides a graphical layout editor which allows developers to create application UI (user interface) using a simple drag and drop interface.

Chapter 2

State of Android SDK installation procedure on (Debian) Linux

You could say that Android SDK installation procedure isn't done in automatic manner on any of today's leading operating systems (Linux, Mac OS X and Windows). However on Mac OS X and Windows this installation procedure is done in a somewhat user friendlier environment while Linux lacks one such procedure which is mostly due to the environment itself rather than the installation procedure of the components itself.

If we are talking about regular or even freshly installed Debian machine, even before getting to part of installing Android SDK we should install a set of prerequisite software such as: "*openjdk-6-jdk*" which is OpenJDK Development Kit and is a open source and free implementation of Java programming language. This is standard for Java related things on Debian, even though we could also use Oracle Java JDK package whose official support has been dropped by Debian even from the "non-free" repositories.

It's also important noting we could also go with "*openjdk-7-jdk*" package, a latest major Java 7 update which was released in the end of 2011, but since version 6 is still the official Java version in Debian we'll be relying on this version. Moving to version 7 could be planned for some of the future versions. Another mandatory package which is required to install is "*eclipse*" which consists of Eclipse IDE, an Extensible Tool Platform and Java IDE. Also please note that up until this point installation procedure doesn't differ much then on the other operating systems we've mentioned.

In case user is running a 64 bit architecture system he must install “*ia32-libs*” a package which contains runtime libraries for the ia32/i386 architecture configured and ported for use on 64 bit Debian kernels such amd64 or ia64. Simply put this package allows you to run 32 bit applications on a 64 bit platform. Another package a 64 bit user should install is “*lib32stdc++6*” which contains additional GNU Standard runtime library for C++ programs that were built with GNU compiler (GCC).

Optional list of additional applications that could be installed are:

- git - popular version control system designed to handle very large projects and efficiency (fast, scalable, distributed revision control system)
- ant - Java based build tool like make
- libxml2 - GNOME XML library
- libxml2-dev - Development files for the GNOME XML library

Next step would be to download Android SDK package from official Google Android Developers website and unzip it, once uncompressed Android SDK will provide us with a new “*android-sdk-linux*” directory, hypothetically speaking if this directory is located in our *\$HOME* directory; on Linux */home* directory is place where every user keeps its own private data, and every user gets its own space under */home* directory depending on their username.

So let’s say our user “John Doe” has username which is “*johndoe*”, under Linux filesystem *\$HOME* directory for user “*johndoe*” will point to */home/johndoe*. *\$HOME* is a symbolic link to current user’s home directory. In order for “*johndoe*” to run “*android*” command instead of having to specify full path to where that application resides we'll have to add *\$HOME/android-sdk-linux/tools* and *\$HOME/android-sdk-linux/platforms* directories to our PATH

- `PATH` is environmental variable that tells shell which directories to search for executable files in response to commands issued by a user.
- `.profile` can contain a series of commands or settings that bash allows you to executes automatically once the user logs in. We added this to `/etc/profile.d/` which is system wide profile for *Bourne* compatible shells including *bash*.

Besides user having to know the location of all the files he needs to edit, and have a general knowledge of tools he needs to use. To demonstrate how exhausting this whole process was before “android-sdk-linux” we’ll demonstrate how this procedure was done manually:

- We will simply add this with our favorite text editor, in this case I’ll use “*vim*”

```
vim .profile
```
- add following to the end of the file:

```
# android sdk  
  
PATH="$HOME/android-sdk-linux/tools:$HOME/android-sdk-  
linux/platform-tools:$PATH"
```
- First line is a comment which tells me what the line after it does, while the second line adds executable files from following directories to be run in shell without providing its full location.
- To make changes effective without logging out and logging in run:

```
export PATH="$HOME/android-sdk-linux/tools:$HOME/android-  
sdk-linux/platform-tools:$PATH"
```

This will allow us to run “*android*” (Android SDK and AVD Manager) (which is located in *\$HOME/android-sdk-linux/tools/android* from shell directly instead of having to specify its full location in order to run it.

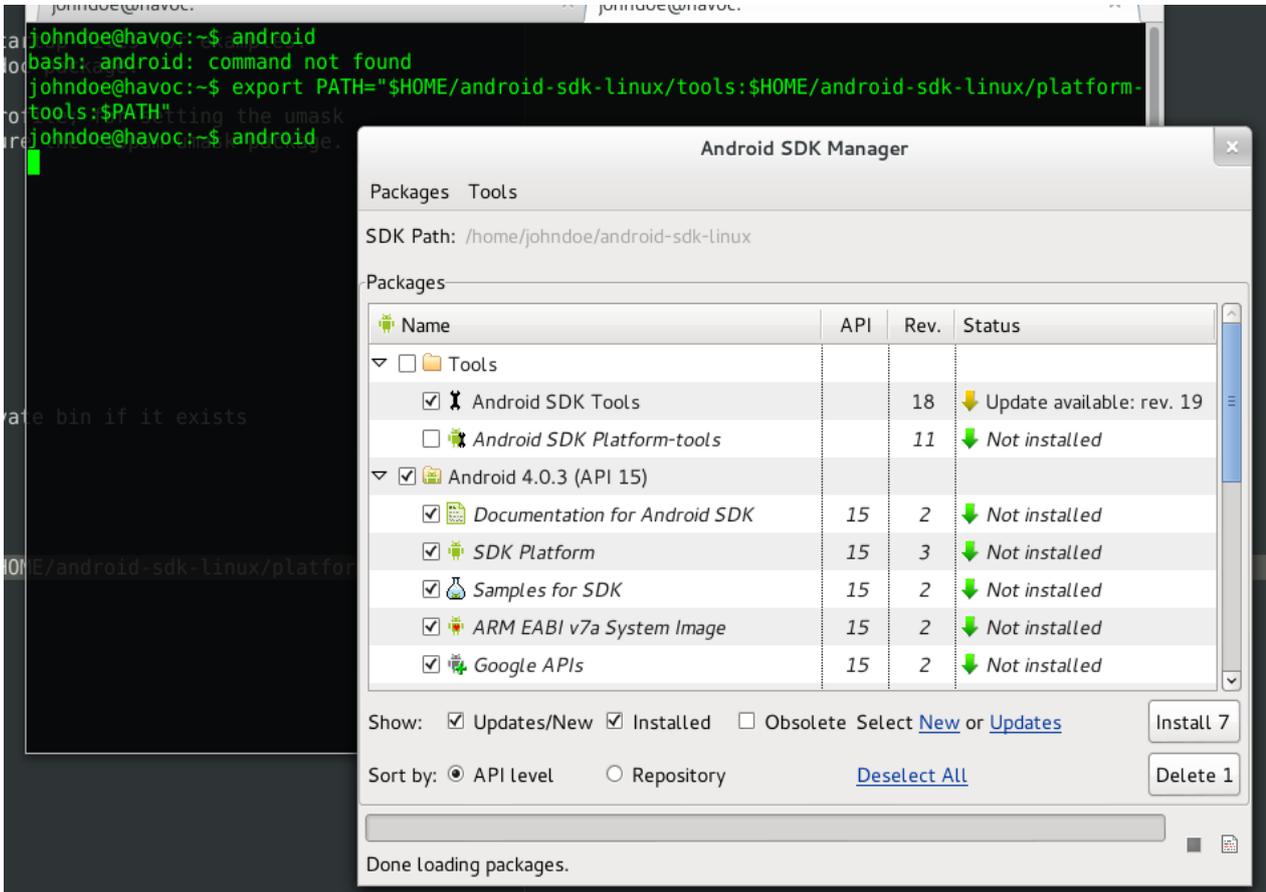


Figure 2: Result of running "android" command in terminal before and after adding PATH

Next step is the standard installation procedure as we can see it on Windows and Mac OS X, where it consists of selecting “Available packages” and “Android Repository” which provides you with packages of different Android API and Platform Tools revisions. Once the desired package/s is/are selected it's enough to mark “Accept” followed with “Install”.

It is highly recommended to install “Android SDK Tools” and “Android SDK Platform-tools” and at least one API level, with Android 2.1 being used on 97% of Android devices.

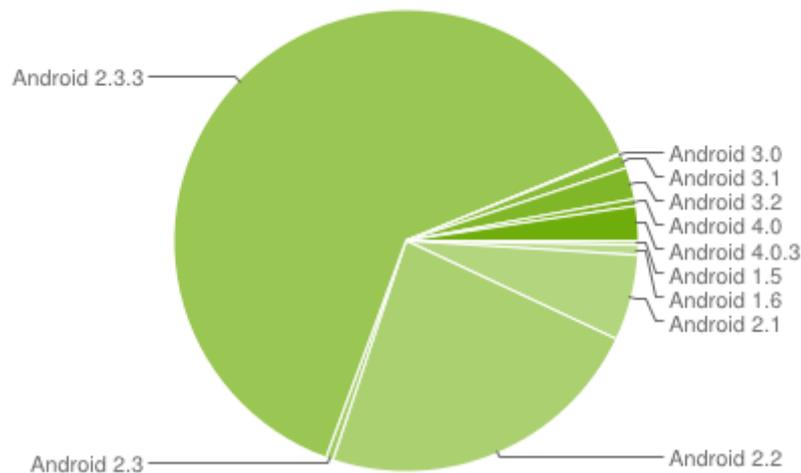


Figure 3: Android platform version distribution according to [Android Developers](#)

Upon successful installation if ADB (Android Debug Bridge) requires to be restarted, answer affirmatively. This step completes the Android SDK installation.

Eclipse ADT plug-in installation

- In order to install eclipse ADT plug-in, it's first required to run Eclipse after which you go to “Help” > “Install New Software”
- On next window “Available Software” click the “Add” button and add following:
 - Name: ADT Plug-in
 - Location: <https://dl-ssl.google.com/android/eclipse/>
- Now back in “Available Software” window you should see “Developer Tools”, select it and proceed to next screen by clicking the “Next” button.
- In next window you'll see details in which Android DDMS, Android Development Tools, Android Hierarchy Viewer, Android Traceview will be installed, proceed by clicking the “Next” button.

- On Next screen in order to proceed you'll need to accept the license agreement.
- Next screen will lead you through packages installation and integration into Eclipse.
- Upon message about unsigned content (Google certificate) it's necessary to accept it and click on “Ok” button in order to proceed.
- Upon successful installation Eclipse Software Update dialog will encourage you to restart Eclipse in order to apply changes, restart in our case is required. Click “Yes” to restart.
- Eclipse ADT plug-in is now installed and integrated into Eclipse, however after Eclipse has been installed you'll be greeted with this dialog:



Figure 4: Screen we get after successful ADT plug-in installation

- This is a fairly new addition to ADT plug-in after it has been installed, in past you had to manually select SDK location which means you'd have to install SDK itself

first. Now you're able to select option to “Install new SDK” or select to “Use existing SDKs” which is step we'll use in our case since we already had our Android SDK installed and set up.

- After locating where your “android-sdk-linux” is located click the “Next” button.

After this step, your Eclipse ADT Plug-in is successfully installed and setup, at which point you're ready to start developing your Android applications. This “tutorial” portrays what Android SDK installation procedure seems like on Debian and generally any other modern Linux distribution. Following a “tutorial” like this one may not be a problem for some, but is

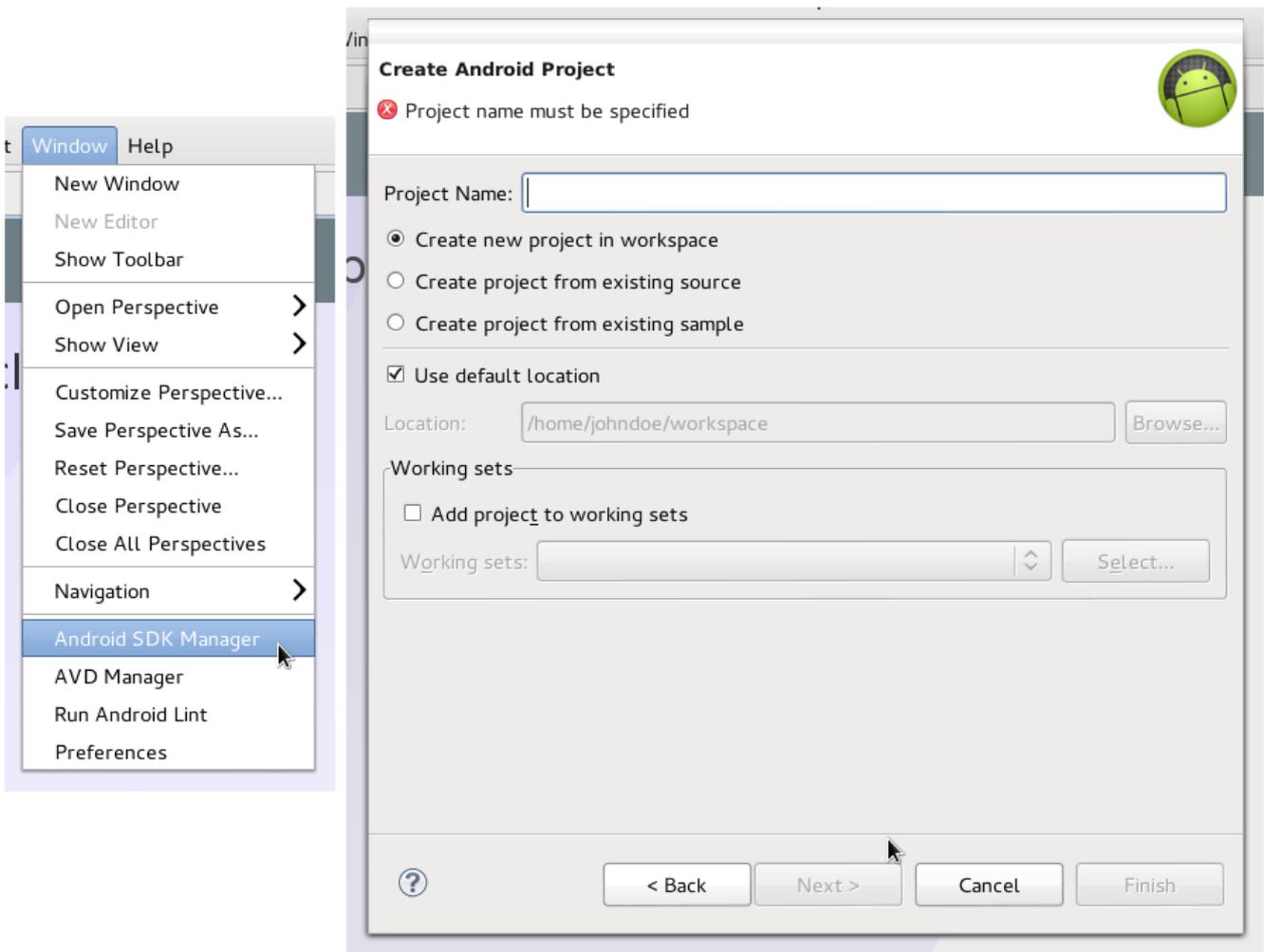


Figure 5: New features inside of Eclipse as result of successful ADT install

it really necessary in order to be ready to start developing Android applications on your favorite Linux distribution?

Even besides all of this, this “tutorial” only covered installation of Android SDK and ADT Plug-in without installing full support for your mobile device, nor adding support for MTP which is also part of the “android-sdk-installer” application.

Explain how this process differs on Windows, Mac OS X and Linux

Even though Windows and Mac OS X may seem as more friendly operating systems, installation of Android SDK along with Eclipse ADT plug-in doesn't differ so much from one on Linux, especially if we compare it to Mac OS X which is based on Darwin a project based on BSD, thus with both of them having common UNIX roots.

On Windows this procedure can be divided into surprisingly similar steps that we can see on Linux:

- Getting Android SDK package
- Extracting it and setting environment variable PATH to the location of SDK
- Setting it a environment variable PATH will do almost the same thing we see on Linux, where when command “android” or “emulator” is run in Windows Command Prompt will launch Android SDK Manager, or an Android Emulator.
- Installing JDK (Java Development Kit)
- Since JRE (Java Runtime Environment) won't be enough it'll be required to install JDK (Java Development Kit) which includes JRE.
- Install Eclipse
- Once JDK/JRE has been installed user will be able to install and run Eclipse
- Eclipse ADT Plug-in

This step is identical to one we saw on Linux

While we may say Windows and Linux Android SDK installation procedure is different, installation on Mac OS X is almost identical to one on Linux. Coming from the same Unix roots, updating PATH environment variable will be done in same manner it's done on Linux, by updating `~/.bash_profile` with default PATH variable.

- Eclipse and Java is obtained from their respected sites and installed in similar manner, while Eclipse ADT Plug-in is installed just as it's installed on any of these platforms.
- Installing device support and getting it recognized by the system

Windows always had problems with device drivers, however luckily for Windows users drivers were usually issued by manufacturers and are installed in semi-automatic mode. Android devices should be automatically detected by the system, in case they are not even with the help with drivers from Android SDK you can always rely on [Google USB Driver](#) or in form of an [OEM USB Drivers](#)

Mac OS X users/developers can skip this part as their drivers are being automatically installed. Automatic device recognition is usually the case on Linux as well, and Linux user can usually skip this part as well, but in order to add full support for specific devices “udev” (Linux kernel device manager) rules need to be manually specified in order for device to be fully recognized.

- In these rules each device manufacturer is identified by a unique vendor ID, as specified `ATTR{idVendor}` property.
- Example of one such rule for: Google Nexus S phone looks like:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666",  
OWNER="plugdev"
```

- MODE in this case represents a read/write permissions the same way they are used in “*chmod*” (Unix command which tells users how much access they have to certain file) fashion.
- OWNER represents a group to which a user has access to, or is part of. Interesting peculiarity regarding this is that many distributions (Red Hat based) have “usb” group while Debian derived distributions refer to that group as “plugdev”.
- One can see to which groups he belongs by running “*id*” command in the terminal.
- These rules are usually written in a filename called “*51-android.rules*” and placed in `/etc/udev/rules.d/`
- They are given both read permissions for all users, groups and others (*chmod a+r*)

Part of “android-sdk-installer” application is to have function for user to automatically add these set of rules without user having to bother with any of it, instead of him having to manually edit *udev* rules, having to know his device USB vendor ID, or what each group is used for and so on.

However, after all of this has been said, even though Linux might have scored a lot of points in “complicated” category, it has one great benefit over its competitors which is its software repositories. Unlike Mac OS X or Windows where you have to get all the components “manually” from Eclipse to JDK in separate locations, from even separate mediums a big advantage of Linux ahead of these systems are its software repositories where all of this software can be downloaded and installed in click of a mouse.

Once “android-sdk-installer” is fully completed application it'll find its place in these repositories and will be able to install all the needed components from a single place with single click of a mouse.

How the idea of an “android-sdk-installer” was born?

A perfect example could be Google's recent migration from USB mass storage protocol to the MTP (Media Transfer Protocol) with latest Android version 4.0 “Ice Cream Sandwich”. I was reading one of my colleagues article on “[\[How to\] Connect your Android Ice Cream Sandwich Phone to Ubuntu for File Access](#)” and I was mesmerized, not because of the quality of his article but rather because I previously did the same thing in order to get MTP enabled on my system and this article gave me ability to read and see the whole situation from another angle and in all honesty to see how complicated this whole process was.

However, this just came in as another piece of the puzzle as idea of having an “Android SDK” installer was present in my mind for some time, idea of having Android SDK along with Eclipse ADT Plug-in seamlessly integrated into Debian Linux. Peculiarities like this one, only extended my idea of this “installer” giving it extra features such as “add hardware device support for Android devices” and ability to “add MTP support (enables USB mass storage support on \geq Android 4.0 ICS)”.

From a technical point of view, it was only extending current installer possibilities and adding couple of additional packages. Of course I'll explain process of my plans of integrating every component later on in chapters dedicated to that purpose. But generally speaking even though this idea was in my mind for a long time, this was probably the event that „sparked“ it the most and after which all the “pieces of puzzle” came in together.

What kind of advantage does this bring to Linux?

This will give an almost incomparable advantage to Linux ahead of its competitors Windows and Mac OS X, in “Explain how this process differs on Windows, Mac OS X and Linux” we saw that when it comes to complexity of this task Linux is not that much behind its competitors, from a given aspect and point of view it could even be in an advantage from a very start.

But advantage Linux has and which can be measured with its competitors is its platform architecture and with certain degree of knowledge put all these components together and make another application out of it which automatically takes care it all, from a user/developer all is required to do is give their input on few very simple prompts.

After release of one such installer, which I'm planning to release under one of the Open Source licenses will encourage developers coming for other platforms to come up with versions of same installer for Mac OS X and Windows, as frankly speaking with code as well as the documentation available for some of the prospective developers changes that need to be made in order to make it work on their platform are minimal.

In general I think one such installer will give Linux great advantage over other platforms when it comes to Android developers installing a simple package and straight after that being able to developer for their favorite platform without having to get their “hands dirty” or losing any of the valuable time in the process. Even if tomorrow the same package was delivered for other platforms, Linux's advantage over all of them would be that this package would be available online on one of its software repositories. In the end it would allow it to break all the “myths” of it being complicated and strictly developer/hacker oriented system.

Application Design and components function

Android is the biggest and fastest growing mobile operating system based on Linux, but even besides this fact support for devices running it as well as the development on this platform (Linux) is far from ideal. It's important to note that Linux is fully and officially supported, with Google and Open Handset Alliance providing all the necessary tools and necessary documentation.

However there's a gap in this whole process, gap which constrains users or/and developers to complete the installation process in a error free and semi-automated manner by selecting which component they would like to have installed and fully working with a single hit of a button.

Goal of this project is implementation of an installer which will install/setup selected components in semi-automated manner. Installer consists of few parts which are:

- Implementation and integration of Android SDK (Software Development Kit)
- Ability to install hardware device support for all Android devices
- Installation of ADT (Android Developers Tool) plug-in for Eclipse IDE
- Adding support for MTP (Media Transfer Protocol)

1. SDK – Android SDK (Software Development Kit) installer

Function of this component is to install Android SDK by making it available for all users on the system, besides bare installation, its function is also to configure and sets the path for the users so user can get Android SDK with a single affirmative answer given to the installer. This will allow users to run applications such as “*android*” (Android SDK

Manager), “*emulator*” (Android mobile device Emulator), “*adb*” (Android Debug Bridge) and rest of the tools and applications that are included with Android SDK package.

2. ADT – Installs ADT (Android Developers Tools) plug-in for Eclipse IDE

Even with having SDK installed, what gives the Android developer the real power and ability to develop, build and deliver Android applications within an integrated environment is ADT (Android Developers Tools) plug-in for Eclipse IDE. This component is supposed to install the plug-in the shorter way, without user having to go through all the steps manually, what user sees is a single question while the installer should do the rest of the process without bothering user while doing it. As with SDK, plug-in will be configured and ready to use along with path link set to Android SDK location, ready to use out of box.

3. Driver – adds support/install drivers for certain mobile devices

In one of the earlier chapters it was mentioned how for Linux and Mac OS X since they come from same background (Unix), installation of drivers is usually unnecessary step as they are automatically recognized and installed. Hardware device support for devices and manufacturers is implemented into the Kernel itself and thus they are automatically recognized. However, with so many different Android devices and manufacturers this isn't quite the case, and “*udev*” (Linux device manager) rules need to be adjusted in order for everything to function properly.

- For Google devices one such rule would look like:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666",  
OWNER="plugdev"
```

- while for Samsung devices same rule would be:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", MODE="0666",
```

```
OWNER="plugdev"
```

- Sony Ericsson:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0fce", MODE="0666",  
OWNER="plugdev"
```

and et cetera, what this component does is eliminates users need to fiddle and create *udev* rules, along with specific information that is required by his devices manufacturer, not to mention that every single of these attributes means something else and could possibly harm your system if not handled properly. That's why this installer allows the user to install support for his device by simply affirmatively answering the questions.

4. MTP – Adds MTP support

As mentioned it was shown and explained how difficult one such seemingly simple step can turn out to be, selection to install this component will take care of everything for the user automatically. Up until latest Android version Android v4.0 “*Ice Cream Sandwich*” your Android phone was connected to your computer using USB Mass Storage (UMS) so once connected your device would appear on your system just as your USB flash drive would appear.

This seemed to work on Linux and Windows without any problems or any additional installations, while on Mac you would use a “[Android File Transfer](#)”, however with latest Android version and their latest device Galaxy Nexus, Google decided to leave USB Mass Storage (UMS) over Media Transfer Protocol (MTP).

Google stated they are leaving UMS over MTP because up until *Galaxy Nexus* every Android device had two partitions, internal one where system files and installed applications

were stored and another partition which would server for your private data, such as photos, music, videos and other media. This second partition was usually used as USB mass storage device it was done this way because USB mass storage protocol is a block-level protocol, meaning that partition couldn't be mounted on two different systems at the same time. Simply said, in order to have one partition mounted on your computer you would have to unmount it from your phone first.

The problem with UMS was that your “internal storage” would fill up, while your “external” storage would have gigabytes of free space for storage of that same data, that's why in more recent versions of Android you had an option where you could move and/or store application data on your “external” storage.

What MTP offers is ability to have everything stored on a single partition, for example: partition wouldn't have to be unmounted from phone in order to be mounted on your computer meaning that most of the applications would be left in operational state while you're transferring your private media or other types of data.

On a long run this makes things much less complicated, user doesn't have to care about size of his internal or external partition as it's all getting sorted under one partition. Another “side affect” of this move is that Android can now use *EXT* file system (which is also a default filesystem for most Linux distributions) instead of having to rely on *FAT32* as a filesystem for the partition that was used as “USB massive storage”. Besides *Galaxy Nexus*, it's most likely that most manufacturers will adapt to this model as they adapt to the latest Android version.

Since Windows Vista MTP support is built in into the system, while Windows XP users need to install Windows Media Player 10 and higher in order to get support for MTP. Linux and Mac have software packages which provide these systems with MTP support.

However, to have MTP support installed on Linux, even though there are software packages that provide it with MTP support, the whole process is much more than just installing packages and it also involves editing *udev* rules and configuring *fuse* (Filesystem in Userspace). Having this component will simplify this tiresome process into a single hit of a key.

Define: “android-sdk-installer”

Even with full support from Google for Android SDK and its great documentation, installation of SDK isn't completed in an error-free nor automated manner. It's done through a “step by step” process with users/developers usually relying on various tutorials in order to achieve this goal. For thousands of new Linux users who are Android developers, meddling with Linux terminal and its internals on their first day in order to have their Android SDK working. Even for more experienced users this process doesn't sound too inviting, and this installer comes in to address those issues in the most simple and automated manner.

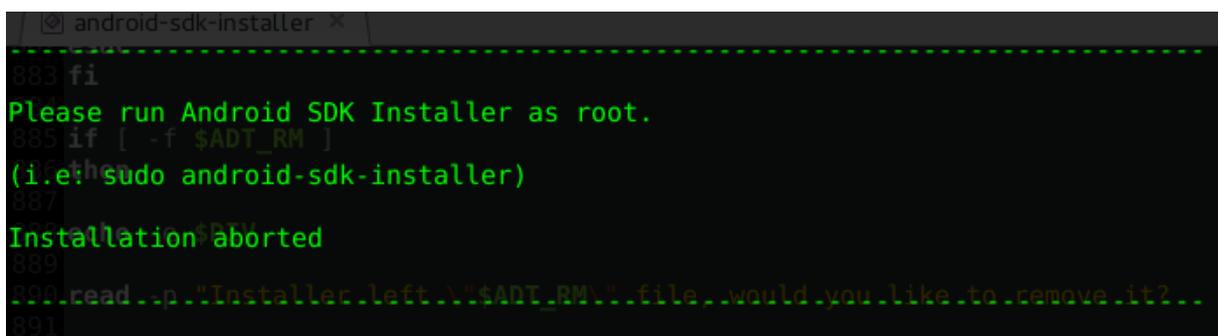
“android-sdk-installer” is imagined as a utility which allows user to install and configure Android SDK, Eclipse ADT Plug-in, add hardware device support for the Android-based devices as well as having features such as “install MTP support”, put in most simple jargon this is an application which allows you to install all the needed components in the most simple manner so one can approach to development of Android applications within minutes.

Painful state of the installation procedure was perfectly portrayed in the beginning of the chapter in “State of Android SDK installation procedure on (Debian) Linux” and “Eclipse ADT plug-in installation” and please be aware that even though this is the very first version of “android-sdk-linux” all of the mentioned issues and the wanted components are implemented and working.

Please note, in following part I'll explain how certain issues and components were handled and will be solely referring to *Appendix: android-sdk-installer*. This installer is imagined to have a “waterfall” flow in which it checks the system state and/or presents the state of required tools and based on those parameters asks the user for his input in case action is required. At the same time all the steps are offered option to “Skip” to the next component as in case of running the installer again in order to reconfigure some of the components which weren't previously configured.

First part of the code is installer's description, its copyright and license (Appendix: Lines 3-20). This is very important as this installer was written in *bash* which is available on all modern Linux distributions, tomorrow once it's publically published due to its license (GPL) developers from other Linux distributions will be able to use this code and tailor this installer to their own distributions needs which means it won't be solely restricted to Debian and its derivatives but it will rather be available for all the distributions out there.

[Appendix: Lines 28-43] check if the user has “root” privileges otherwise the installer procedure cannot continue, this is due to the fact that almost every single one of these components and/or issues requires root privileges in order to address their resolution thus the reason installer won't be able to continue without same set or privileges. Sometime in future I'm planning to implement support for users with “regular” set of permissions to use the applications components which don't require root privileges.



```
android-sdk-installer x
...fi
Please run Android SDK Installer as root.
...if [ -f $ADT_RM ]
(i.e: sudo android-sdk-installer)
...then
Installation aborted
...read -p "Installer left ..$ADT_RM..file..would you like to remove it?.."
...fi
```

Figure 6: refernece to Appendix: Lines 28-43

[Appendix: line 45-91] addresses the issue of 64 bit architecture and dependencies it requires, this part requires more work which will be seen in future versions.

[Appendix: line 93-226] deals with process of doing any pre-cleaning if necessary and setting up the necessary environment that's required for our installer. In this stage, if any of the files which installer will use are found in the same directory (during component download) that component will automatically make a copy of the same file thus leading to errors. This is also planned to be addressed in one of the future versions of the installer, where in case one of the components are found in that same directory, installer will be able to use it.

[Appendix: Lines 228-268] in case */opt/android-sdk-linux* directory already exist, installer will offer to make a copy of that directory as it will also ask for permissions to use that directory and overwrite it with new installation data. In case this directory doesn't exist installer will create it and give it right set of permissions for all system users to use.

[Appendix: Lines 272-315] in this step installer detects whether the directory where Android SDK Tools are stored is empty, if this is the case all of the system would be left without the “core” Android SDK tools such as android (Android SDK and AVD Manager), emulator (Android Emulator), ddms (Dalvik Debug Monitor) and et cetera. In case it is empty, installer will offer to download “Android SDK Starter Package” which contains all of the mentioned tools and install them.

```

-----
883 fi
Welcome to Android SDK Installer 0.1
885 if [ -f $ADT_RM ]
The installer will guide you through the rest of the process.
887
888 echo -e $DTV
-----
889
890 read -p "Installer left \"$ADT_RM\" file, would you like to remove it?
x86_64 architecture detected and its dependencies are satisfied
892 [Y]es, [N]o, [S]kip : " response
893
900 case $response in
-----
905
Install Android SDK Starter Package?
897 rm -f $ADT_RM 2>/dev/null
Provides: tools such as "android" (Android SDK and AVD Manager),
"emulator" (Android Emulator), "ddms" (Dalvik Debug Monitor), et cetera ...
900 [Nn]* )
[Y]es, [N]o, [S]kip: y
902 ;;

```

Figure 7: reference to Appendix: Lines 272-315

```

-----
883 fi
Welcome to Android SDK Installer 0.1
885 if [ -f $ADT_RM ]
The installer will guide you through the rest of the process.
887
888 echo -e $DTV
-----
889
890 read -p "Installer left \"$ADT_RM\" file, would you like to remove it?
x86_64 architecture detected and its dependencies are satisfied
892 [Y]es, [N]o, [S]kip : " response
893
900 case $response in
-----
905
Install Android SDK Starter Package?
897 rm -f $ADT_RM 2>/dev/null
Provides: tools such as "android" (Android SDK and AVD Manager),
"emulator" (Android Emulator), "ddms" (Dalvik Debug Monitor), et cetera ...
900 [Nn]* )
[Y]es, [N]o, [S]kip: y
902 ;;

Will now download the Android SDK Starter Package
904 [Ss]* )
--2012-05-30 21:03:47-- http://dl.google.com/android/android-sdk_r18-linux.tgz
Resolving dl.google.com (dl.google.com)... 74.125.232.228, 74.125.232.229, 74.
Connecting to dl.google.com (dl.google.com)|74.125.232.228|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 29731463(28M) [application/x-tar]
Saving to: 'android-sdk_r18-linux.tgz'
911
93% [=====> ] 27,814,332 815K/s eta 3s
913 esac

```

Figure 8: "Android Starter Package" installation procedure

[Appendix: lines 318-380] in this step, installer offers to install latest Android SDK API Platform which supports all the latest features and to install Android API 2.1 which is supported on 97% phones and tablets. It's also important to note once the installer is finished, in case none of these platforms was installed “*android*” (Android SDK and AVD Manager) allows the user to install the mentioned API's as well as all the other API's to this date. In case they were installed they will be recognized to be in such state.

```

android-sdk-linux/tools/support/
android-sdk-linux/tools/support/annotations.jar
android-sdk-linux/tools/adb_has_moved.txt
android-sdk-linux/tools/draw9patch
android-sdk-linux/tools/etc1tool
android-sdk-linux/tools/mksdcard
android-sdk-linux/tools/traceview
android-sdk-linux/tools/emulator-arm
014 fi
Android SDK Starter Package successfully installed
016 echo -e $DIV
017 -----
018 echo -e "Installation complete. Feel free to run \"android-sdk-installer
Install latest Android API\4.0.3 API\15; revision 3?
(supports all the latest features)
021 exit 0
[Y]es, [N]o, [S]kip : █

```

Figure 9: Installer offers to install the latest available Android API

```

extracting: /opt/android-sdk-linux/platforms/android-4.0.4/templates/ic_launcher_mdpi.png
extracting: /opt/android-sdk-linux/platforms/android-4.0.4/templates/ic_launcher_hdpi.png
extracting: /opt/android-sdk-linux/platforms/android-4.0.4/templates/ic_launcher_ldpi.png
015
Successfully installed Android API 4.0.3 API 15, revision 3
017
018 echo -e "Installation complete.. Feel free to run \"android.sdk.installer
019 time to (re)configure.\n\nEnjoy!\n";
Install Android API 2.1, revision 03? (supported by ~97% phones and tablets)
021 exit 0
[Y]es, [N]o, [S]kip: █

```

Figure 10: Installer offers to install Android 2.1 API

[Appendix: lines 384-472] installer offers to install and configure PATH environmental variable which allows all the system users to run tools such as *android*, *emulator*, *ddms* and et cetera directly from their terminal . In one of the previous parts in

which installation state was compared to Windows and Mac OS X we'll clearly see what kind of advantage this gives even ahead of Windows in which this step has to be performed manually.

```

extracting: /opt/android-sdk-linux/platforms/android-2.1_r03-linux/templates
icon_mdpi.png
inflating: /opt/android-sdk-linux/platforms/android-2.1_r03-linux/templates
java_tests_file.template
.....
Successfully installed Android API
.....
Configure PATH environmental variable?
.....
Will allow you to run tools such as "android" (Android SDK and AVD Manager),
"emulator" (Android Emulator), "ddms" (Dalvik Debug Monitor) directly from
Terminal.
.....
[Y]es, [N]o, [S]kip: y

```

Figure 11: Installer offers to configure the PATH environmental variable

[Appendix: lines 474-533] installer handles the installation of Android ADT plug-in and Eclipse in case it hasn't been installed. However this components has the number 1 priority when it comes to installer future releases, where it is planned for this component to be installed in a form of a Eclipse plug-in rather than installing it by “brute force” which was the standard of plug-in installation up until Eclipse 3.3. However with Eclipse 3.8 release around the corner it is necessary to tailor this component to its present standards.

```

.....
010 echo -e $DIV
Install ADT (Android Development Tools) Plugin for Eclipse IDE?
010 echo -e "Installation complete. Feel free to run \android-sdk-installer
[Y]es, [N]o, [S]kip: y
.....
Eclipse IDE is NOT installed, would you like to install it?
[Y]es, [N]o, [S]kip :y

```

Figure 12: Start of ADT Plug-in installation procedure

```

-----
892 [Y]es, [N]o, [S]kip : " response
Install ADT (Android Development Tools) Plugin for Eclipse IDE?
894 case $response in
[Y]es, [N]o, [S]kip: y
896     [Yy]* )
Eclipse IDE is NOT installed, would you like to install it?
[Y]es, [N]o, [S]kip :y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
 eclipse
0 upgraded, 1 newly installed, 0 to remove and 10 not upgraded.
Need to get 0 B/54.3 kB of archives.
After this operation, 105 kB of additional disk space will be used.
Selecting previously unselected package eclipse.
(Reading database ... 149032 files and directories currently installed.)
Unpacking eclipse (from ./eclipse_3.7.2-1_all.deb) ...
Setting up eclipse (3.7.2-1) ...

Okay, will now download Eclipse ADT Plugin
2012-05-30 21:08:09-- http://dl.google.com/android/ADT-18.0.0.zip
Resolving dl.google.com (dl.google.com)... 173.194.39.64, 173.194.39.65, 173.
94.39.66, -e..$DIV
Connecting to dl.google.com (dl.google.com)|173.194.39.64|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12834793 (12M) [application/zip]
Saving to: `ADT-18.0.0.zip'
021 exit 0
66% [=====>] 8,492,244 767K/s eta 8s

```

Figure 13: ADT Plug-in installation

[Appendix: Lines 535-722] in these 187 lines installer offers to install hardware device support for all the Android devices. It also covers many scenarios in which it also recognizes if such rules already exists, in final case it will add hardware support for all the devices in accordance to Google's latest "hardware devices list".

```

 inflating: /tmp/plugins/overlay.com.android.ide.eclipse.adt.overlay_18.0.0.
201203301601-306762.jar
 inflating: /tmp/site.xml
 creating: /tmp/web/
 inflating: /tmp/web/site.css
 inflating: /tmp/web/site.xml
117
118 echo ..e "Installation complete.. Feel free to run \\.android.sdk.installer
119 time to (re)configure.\n\nEnjoy!\n";
Add hardware support for you Android devices?
021 exit 0
[Y]es, [N]o, [S]kip :

```

Figure 14: reference to Appendix: Lines 535-722

[Appendix: Lines 724-782] installer adds Add MTP support which enables USB mass storage support on Android 4.0 and later, this component has number 2 priority position on future installer releases as at the time of writing this installer and paper I was not in possession of one such device (Google Galaxy Nexus) to test it out properly.

```

004 inflating: /tmp/web/site.css
005 inflating:*/tmp/web/site.xml
006 chmod 666 $ADT_RM
-----
007
Add hardware support for you Android devices?
009 echo "Wrong value: installaton aborted."
[Y]es, [N]o, [S]kip :y
Adding hardware support
012
Hardware device support successfully installed
014 fi
015
-----
016 echo -e $DIV
Add MTP support? (enables USB mass storage support on Android >= 4.0)
018 echo -e "Installation complete. Feel free to run \"android-sdk-installer
[Y]es, [N]o, ([S]kip):y
020
Packages seem to be missing, install required packages?
[Y]es, [N]o :y
    
```

Figure 15: reference to Appendix: Lines 724-782

[Appendix: 784-914] as in the beginning of this process, installer will offer to clean up and remove any of the unnecessary files. Since files were created while using root permissions, even if user decides to skip installer will leave these files with right set of permissions for ordinary user to deal later on.

```

MTP support successfully installed
010 exit 1
-----
012
Installer left "android-sdk_r18-linux.tgz" file, would you like to remove it?
014 fi
[Y]es, [N]o, [S]kip :y
016 echo -e $DIV
-----
018 echo -e "Installation complete. Feel free to run \"android-sdk-installer
Installer found "android-15_r03.zip" file which might defect rest of
the installation process, remove?
021 exit 0
[Y]es, [N]o, [S]kip :y
    
```

Figure 16: Reference to cleaning up process

[Appendix: Lines 918:921] installer prints out the final message and how the user can start the installer in case he needs to (re)configure something.

```

-----
016 echo -e $DIV
Installation complete. Feel free to run "android-sdk-installer" at any
time to (re)configure.
017 echo "Installation complete. Feel free to run \ android-sdk-installer
019 time to (re)configure.\n\nEnjoy!\n";
Enjoy!
021 exit 0
absinthe@havoc:~/Debian/android-sdk-installer/0.1$ █
    
```

Figure 17: Installer goodbye message

This part briefly explained some of the installer through its code, in future besides the number of planned features and options one of the bigger options that's planned is for the installer to get GUI support and be available in GUI.

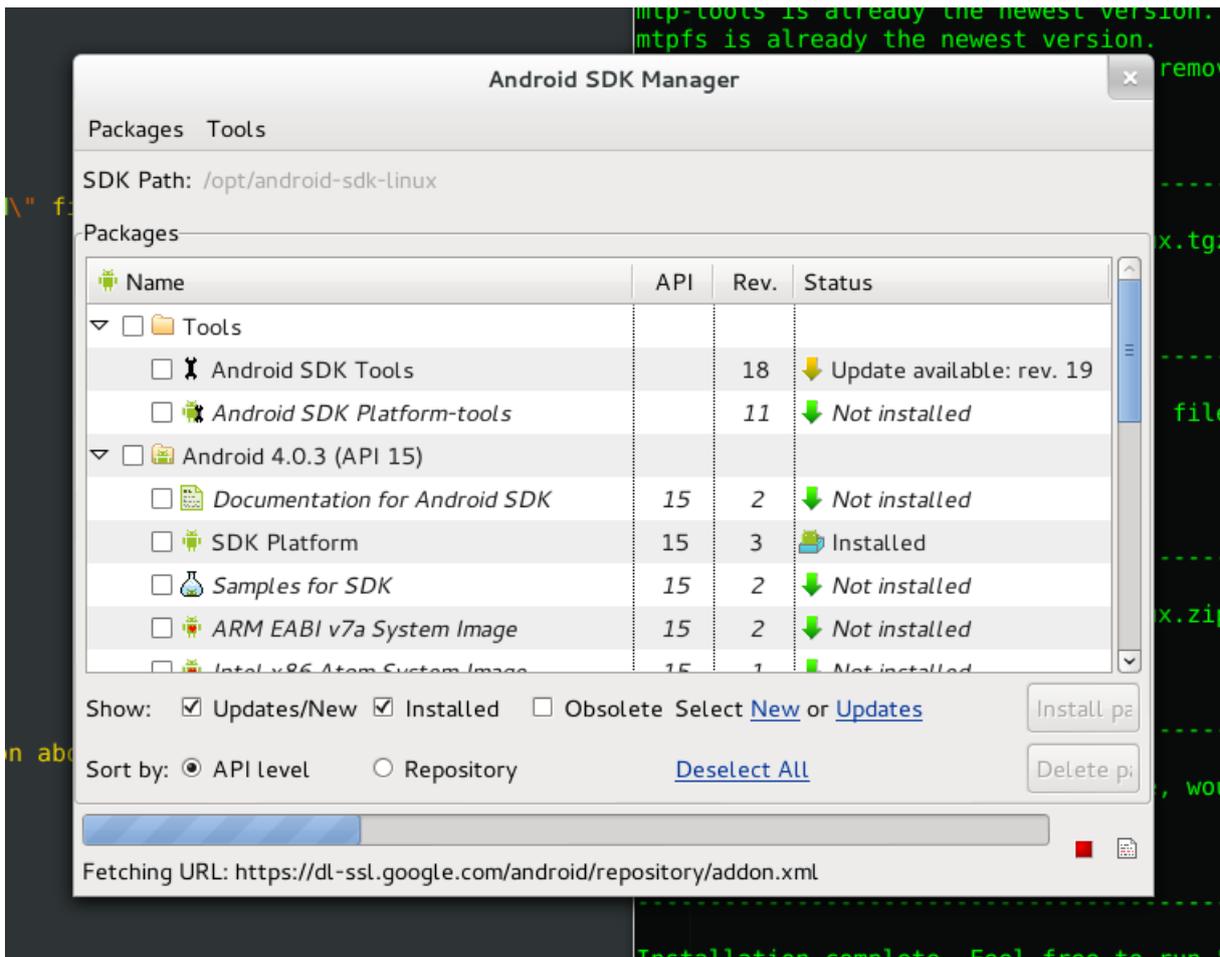


Figure 18: Android SDK Manager running after successful installation

Chapter 3

Which programming languages were used in creation of this application?

For installer script itself main programming/scripting language that was used is *Bash* (Unix command shell), while other components should mostly consist of Java and possibly some C++ and XML.

Whole application should be written as a mixture of *Bash* and features provided by *APT*. Application itself will be written in a single “android-sdk-linux” *Bash* script file which will included in part of “android-sdk-linux” Debian package. The script itself will have all instructions on how to operate once one of the components above has been selected, of course with help of files inside of *debian/* directory manipulation of what needs to be done in combination with the script will be complete. For example, even though we can set dependencies in control file, these will also be set in script itself and will communicate with files within *debian/* in order to make the package.

How was this platform chosen from technical stand point?

Besides Debian being one of the oldest and most influential distributions, another quality which makes Debian a top choice is its Advanced Packaging Tool (APT), which some refer to as one of its biggest and best features due to its speed, dependency control and of course its strict quality control in order to satisfy “Debian's Policy”.

There are many package management tools for various Linux distributions such and each one of them has its own advantages and disadvantages.

The most popular and most widely used ones are APT - (Advanced Packaging Tool [.deb]) and RPM (RPM Package Manager [.rpm]) mostly due to its quality and long history, and as well due to the most Linux distributions being based on those same packaging managers as well as being a derivatives of Debian and Red Hat and them being one of the oldest distributions.

That's why you can see APT being used in most popular distributions today such as Debian, Ubuntu and all its variants and Mint Linux while you can see RPM being used in distributions such as Red Hat, Fedora, CentOS, OpenSuse and Mageia (ex Mandriva, ex Mandrake).

RPM started out as a “Red Hat Package Manager” while today it's referred to as “RPM Package Manager”, however even due to its pedigree most of the time RPM is referred to as being notably slow compared to its competition, and its past was most notoriously known for a term called “RPM dependency hell” which resulted in included software packages version mismatching their dependency on specific version of other software packages. This problem was most active and can relate to early 2000's while today this problem has mainly been sorted out and dependencies are automatically sorted and fixed.

Contrary to RPM problem, our third oldest distribution of today, Slackware never had any “dependency hell” problems, mostly because its package manager (slackpkg) doesn't resolve dependencies between packages. Slackpg will install the desired package (.tgz) and won't check for any software dependencies which usually resulted in application not being able to start forcing you to backtrack which library/ies was/were missing during its launch, in other words everything had to be done manually. Even getting the package itself, unlike APT and RPM you couldn't/can't get package from online repositories rather you have to get it from official sources (Slackware package website). Today there are package managers such as

slapt-get which take care of the repository and dependency problems, but are labeled as third party and are not officially supported.

Package managers of newer generations worth mentioning are Gentoo Linux “*portage*” and Arch Linux “*pacman*”.

Portage's main and most distinctive feature is that once you selected a certain package to install, it will compile that software package from source before installing it, one such mechanism allows it very precise dependency tracking but the toll is taking is time that is required for a certain software package to be compiled from source, especially for some packages such as GNOME or LibreOffice which result in compilation time which is measured in days. However, die hard Gentoo fans will argue that their packages and their system runs faster than the pre-compiled package archives and package software we install on rest of the distributions, even though by some benchmarks this claim don't turn out as true.

Arch Linux “*pacman*” is capable of resolving dependencies and synchronizing package list and even automatically downloading and upgrading (all) packages by running a single command, but what makes it unique is structure of its packages which are compressed into tar archive containing metadata along with its files. Packages are exclusively built using Bash build scripts.

However with all of this said, APT seems to be a package manager with most features, option and with its dependency solving characteristics a truly most reliable and trustworthy solution. It's also worth noting that APT is even capable of handling RPM packages with tools such as “Alien” which allows converting and installing .rpm packages, it can even do the same for Slackware (.tgz) format files.

Tools like “apt-rpm-repository” even provide a feature of porting Debian APT to RPM package formats and creating an APT RPM repository all so RPM based system can be maintained using APT while the repository (APT) itself resides on Debian system.

Even Gentoo diehard fans can be silenced with tool such as “apt-gentoo” which makes Debian fully compatible with these new source-based distributions. With this tool packages can be installed in same way they are installed on Gentoo, where packages are downloaded, their build logs are read and then building process is simulated on their local machine.

Besides numerous features and options APT provides seamless dependency resolution through checking and syncing with online repositories, it does this by relying on */etc/apt/source.list* where online software repositories are defined. APT pinning allows user to download and install packages from other distributions again with seamless dependency resolution, for example: *testing* user wants to install package which is available in *experimental* repository, this can simply be done by issuing:

```
apt-get -t experimental install package-name
```

There are many other APT features which won't be mentioned due to the scope of this paper.

Standard application installation procedure on Linux

Usual way of installing an application on your Linux system without help of package manager is a process which is also known as “compiling and installing from source”. Please note that this procedure can greatly vary from application to its type and numerous other factors, first step is to download the program (usually in compressed format) extract it and into the newly created directory. It also may be worth mentioning that besides Linux this

process is same on any Unix blend, tools involved in this process is GNU Build System also known as *Autotools*.

It is then necessary to invoke *./configure*, this script checks your system libraries and sorts out dependencies that are required in order to build your application. Depending on the application and way that application relates to certain library, but this script will usually either show which dependencies are not satisfied and will continue with the process in case these dependencies aren't mandatory, in case they are your application installation process will stop. Main purpose and job of *./configure* script is to build and create a “*Makefile*” which is necessary file during the installation process as it checks and test configure script to see if all necessary steps and tasks are taken and satisfied in order to compile the software itself.

Upon successful completion *make* is run which automatically builds and compiles all programs code along with its libraries from source code and creates the executable, at this step utility usually doesn't complain about missing dependencies and if there is a sequence it can't go through it'll just break the whole compilation process. Upon successful completion you're ready to install the newly generated executable file/s.

make install is last step which is usually required to be run by a root user. In this step executable files (which were made in make process) are put in developer pre-defined directories in order for them to be successfully executed, this involved make script finding install within *Makefile*. This “final destination” directory is usually a */usr/bin* so the application is allowed to be run by all users on system, but as previously noted this process can greatly vary by application that is being installed.

This installation procedure can greatly vary on different build systems as well applications written in different programming languages, and usually result in many errors and other sort of problems depending from platform to platform. Thus I believe this process

perfectly portrays a job Linux package management system does and to what degree it simplifies and makes installation/removal/upgrade procedure easier and more convenient.

Application creation and installation in Debian

Usual way of application packaging procedure in Debian consists of getting the right set of development scripts, which can all be found in package named “*devscripts*”. Step that usually follows is getting source of upstream program usually found in compressed (tar) format, creating an appropriate environment for the package creation after which *dh_make* is run which prepares original source archive for Debian packaging.

After running “*dh_make*” directory “*debian/*” will be as created in which we'll find many files, rest and all of Debian packaging process will take place within this directory. Most important files and files that are required in order for package to be built are:

control - meta data about package such as its description, architecture it supports, package dependencies, package author and et cetera.

changelog - packages changes history, also worth mentioning is when package is first created, it's specified to which distribution repository its targeting. Also if bug number or (ITP) is labeled that bug is marked for closing. As once uploaded labeled bugs will be automatically marked as fixed in Debian bug tracking system.

rules - this file is what actually creates the package and could be labeled as another “*Makefile*”. Data inside of this file defines how the package will be made, also out of all files in *debian/* this is the only one that's marked as executive.

copyright - file contains copyright and license information for the particular (upstream) package.

Besides these, there are many (optional) files which will be used in case of “android-sdk-installer” package whose function will be described later on. Once we're done setting up files inside of *debian/* we're ready to build and test our package. This can be done many ways, with the simplest one being:

```
dpkg-buildpackage -us -uc
```

dpkg-buildpackage command itself builds the package, while arguments “-us” and “-uc” mean do not sign the source package nor the *.changes* file respectively. While this description itself might bring a lot of ambiguity on what each thing is (*.changes* or what does signing even mean?) this will be explained more in next chapter where implementation of package is described in more detail.

During this package building process, package goes through pretty much the same instruction set we saw in “Standard installation procedure” + it models it to fit Debian APT set. If there are problems during compilation you yourself will need to install the necessary build dependencies and libraries for that package, however upon success built final result is “*package_name.deb*”.

This package can be installed locally using “*dpkg*” (*dpkg -i package_name.deb*) tool which is standard Debian package manager, or using graphical installers such as “*gdebi*” by clicking on the *.deb* file name. Upon installation package will suggest required and optional (suggested) dependencies, these are installed by simply affirmatively answering on that prompt.

Please note that procedure explained above is most basic and simplest way to explain Debian package creation, there are numerous way to create package using different tools such as “*debuild*” or “*git-buildpackage*” and et cetera. “android-sdk-installer” will have much

different and complicated structure then displayed here, which will be seen and available in written form upon its final code release

Application implementation into Debian repository sources

In order to upload your package to one of Debian distribution software repositories you have to satisfy some of the Debian policy requests. First of all you as a future maintainer or perhaps even a developer you must generate and take care of your PGP key, you'll use this key in order to sign your package and/or the changes you made to current package. Since all of the work in and on Debian is done remotely, PGP key is used as form of identification and trust among developers. “Key signing” events are usually held at conferences such as DebConf where developers can exchange their key numbers and with proper form of identification sign them.

Of course, after your package has been made and is even working perfectly, that doesn't mean that it complies with Debian's strict policy requirements, thus after creation before it's uploaded package needs to pass all “*lintian*” errors and even warnings. “*lintian*” is a very powerful package analysis tool which is used for this purpose after package creation. For example you can even run package creation, lintian checkup and do its signing with tools such as “*debuild*” which do this by default.

After your package is created and working properly, has passed “*lintian*” tests and is signed with your PGP key it's a valid candidate to enter Debian software repositories. Until you get a status of a Debian Developer, you'll most likely want to submit your package using “Debian Mentors” website. This website allows you to upload your packages to Debian distribution software repositories, with additional help from current Debian developers which

will guide you through the process or will suggest how to do some of the things you did in your package in a different manner.

Conclusion

In this paper we're greeted with extensive overview of an Android, its SDK, its tools and its state in today's three most popular and used operating systems. Reader is also introduced to what Debian Linux is, and its development model and making a distinction between these two at first similar or even same systems, of course meaning they are both purely Linux based platforms. Yet, due to their licensing model and freedom it allows this could give a great glimpse what you're able to do with these systems if you have right vision, and of course due to its open model how simple it is to achieve this goal.

Of course, idea of “android-sdk-installer” is introduced, and it can perfectly be used to portray the idea that was emphasized in previous paragraphs. It is with power behind the open source idea and community that one such application can even gain advantage over operating systems that are annually fueled with billions of dollars of revenue. However, let that idea not stray you from the original topic this paper introduces which is how to create one such application within Debian Linux, why and how to target which Linux distribution you want your application to work on and to explain whole process of doing so.

In final conclusion this paper should provide the reader with many aspects, from market analysis to problem realization. Through proposition of its solution to detailed creation of the application and its implementation, this in its final form should give a perfect example on how entire community and technology world can benefit from one such idea. On broader scale this topic doesn't just limit on Linux but rather how one such tool with empowered by GPL 2+ license can even provide other operating systems with opportunity to implement same solutions in order to fix its flaws. In general result of this paper besides showing the way from creation of an idea to its implementation, now only how (Debian) Linux can also benefit

even its competitors using Open model and improving technology as a field in general, rather than just a strictly targeted platform.

List of References

Carl Albing, JP Vossen, Cameron Newham. (2007). Bash Cookbook. Sebastol, California:

O'Reilly Media

Mendel Coope. (2011). Advanced Bash-Scripting Guide. Retrieved from:

<http://tldp.org/LDP/abs/abs-guide.pdf>

Steve Parker. (2011). Shell Scripting. Hoboken, New Jersey: Wiley/Wrox

Josip Rodin, Osamu Aoki. (2012). Debian New Maintainers' Guide. Retrieved from:

<http://www.debian.org/doc/manuals/maint-guide/maint-guide.en.pdf>

Developer's Reference Team, Andreas Barth, Adam Di Carlo, Raphaël Hertzog, Lucas

Nussbaum, Christian Schwarz, Ian Jackson. Debian Developer's Reference. Retrieved

from: <http://www.debian.org/doc/manuals/developers-reference/developers-reference.en.pdf>

Martin F. Krafft. (2010). The Debian System, 2nd Edition. München, Germany: O'Reilly

Media.

Michael Kerrisk. (2010). The Linux Programming Interface. San Francisco, California: No

Starch Press.

Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura. (2012). Programming

Android, 2nd edition. Sebastopol, California: O'Reilly Media

Google Inc. (2012). Android Developers. Retrieved from:

<http://developer.android.com/index.html>

IBM Corporation and others. (2005). Welcome to Eclipse. Retrieved from:

<http://archive.eclipse.org/eclipse/downloads/drops/R-3.1-200506271435/org.eclipse.platform.doc.isv.3.1.pdf.zip>

Lars Vogel. (2011). Eclipse Plugin Development - Tutorial for Eclipse 3.4. Retrieved from:

<http://www.vogella.de/articles/EclipsePlugIn/article.html>

Jonathan Corbet. (2011). Bringing Android closer to the mainline. Retrieved from:

<https://lwn.net/Articles/472984/>

Bilal Akhtar. (2011). [How to] Connect your Android Ice Cream Sandwich Phone to Ubuntu

for File Access. Retrieved from: <http://www.omgubuntu.co.uk/2011/12/how-to-connect-your-android-ice-cream-sandwich-phone-to-ubuntu-for-file-access/>

Wikipedia. (2012). Debian. Retrieved from: <http://en.wikipedia.org/wiki/Debian>

Wikipedia. (2012). Android (operating system). Retrieved from:

http://en.wikipedia.org/wiki/Android_os

Steven J. Vaughan-Nichols. (2012). Android and Linux re-merge into one operating system.

Retrieved from: <http://www.zdnet.com/blog/open-source/android-and-linux-re-merge-into-one-operating-system/10625>

Todd Wasserman. (2012). Android Tops 50% Market Share in the U.S. [STUDY]. Retrieved

from: <http://mashable.com/2012/04/04/android-breaks-50-market-share/>

James E. Bromberger. (2012). Debian Wheezy: US\$19 Billion. Your price... FREE!.

Retrieved from: <http://blog.james.rcpt.to/2012/02/13/debian-wheezy-us19-billion-your-price-free/>

Appendix: android-sdk-installer

```

1. #!/bin/bash -e
2. #
3. # android-sdk-installer - Utility to automatically install and configure Android
4. # SDK, Eclipse ADT Plugin, add hardware support for devices as well as add MTP
5. # support.
6. #
7. # Copyright © 2012 Adnan Hodzic <adnan@foolcontrol.org>
8. #
9. # This program is free software: you can redistribute it and/or modify
10. # it under the terms of the GNU General Public License as published by
11. # the Free Software Foundation, either version 3 of the License, or
12. # (at your option) any later version.
13. #
14. # This program is distributed in the hope that it will be useful,
15. # but WITHOUT ANY WARRANTY; without even the implied warranty of
16. # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17. # GNU General Public License for more details.
18. #
19. # You should have received a copy of the GNU General Public License
20. # along with this program. If not, see <http://www.gnu.org/licenses/>.
21.
22. ASI="Android SDK Installer"
23.
24. ASI_VERSION="0.1"
25.
26. DIV="\n-----\n";
27.
28. # Check if user is root or not
29.
30. ROOT_UID=0
31. if [ "$UID" -ne "$ROOT_UID" ]
32. then
33.     echo -e $DIV
34.     echo -e "Please run $ASI as root.
35.     \n(i.e: sudo android-sdk-installer)
36.     \nInstallation aborted"
37.     echo -e $DIV
38.     exit 1
39. else
40.     echo -e $DIV
41.     echo -e "Welcome to $ASI $ASI_VERSION
42.     \nThe installer will guide you through the rest of the process."
43. fi
44.
45. # Detect architecture, if x86_64 install its dependencies

```

```
46.
47. ARCH="$(uname -m)"
48.
49. PACKAGE=ia32-libs
50.
51. #PACKAGE=ia32-libs
52.
53. # "PACKAGE=ia32-libs" needs to be perfected, due to package name (ia32-libs)
54. # because of hyphen (-) in its name results aren't accurate and will report it's
55. # installed even when its not
56.
57. if [ "$(dpkg-architecture -qDEB_BUILD_ARCH)" == "amd64" ]
58. then
59.     if dpkg -s "$PACKAGE" 2>/dev/null 1>/dev/null
60.     then
61.
62.         echo -e $DIV
63.         echo -e "\n$ARCH architecture detected and its dependencies are satisfied\n"
64.     else
65.         echo -e $DIV
66.
67.         read -p "You're running $ARCH arhitecture, there are dependencies
68. ($PACKAGE) which are not present on your sistem? Install?"
69.
70. [Y]es, [N]o, [S]kip: " response
71.
72. case $response in
73.
74. [Yy]* )
75.     echo -e "\nOkay, will now install $PACKAGE\n"
76.     apt-get install $PACKAGE
77.     ;;
78.
79. [Nn]* )
80.     echo -e "\nPlease install $PACKAGE manually"
81.     ;;
82.
83. [Ss]* ) ;;
84.
85. * )
86.     echo "Wrong value: installaton aborted."
87.     exit 1;;
88.
89. esac
90. fi
91. fi
92.
93. # Pre-cleaning process
```

```
94.
95. SDK_R18="android-sdk_r18-linux.tgz"
96. SDK_15="android-15_r03.zip"
97. SDK_21="android-2.1_r03-linux.zip"
98. ADT_RM="ADT-18.0.0.zip"
99.
100.     if [ -f $SDK_R18 ]
101.     then
102.
103.     echo -e $DIV
104.
105.     read -p "Installer found \"$SDK_R18\" file which might defect rest of
106.     the installation process, remove?
107.
108.     [Y]es, [N]o, [S]kip : " response
109.
110.     case $response in
111.
112.         [Yy]* )
113.             rm -f $SDK_R18 2>/dev/null
114.             ;;
115.
116.         [Nn]* )
117.             chmod 666 $SDK_R18
118.             ;;
119.
120.         [Ss]* )
121.             chmod 666 $SDK_R18
122.             ;;
123.
124.         * )
125.             echo "Wrong value: installaton aborted."
126.             exit 1
127.             ;;
128.
129.     esac
130. fi
131.
132.     if [ -f $SDK_15 ]
133.     then
134.
135.     echo -e $DIV
136.
137.     read -p "Installer found \"$SDK_15\" file which might defect rest of
138.     the installation process, remove?
139.
140.     [Y]es, [N]o, [S]kip : " response
141.
```

```
142.     case $response in
143.
144.         [Yy]* )
145.             rm -f $SDK_15 2>/dev/null
146.             ;;
147.
148.         [Nn]* )
149.             chmod 666 $SDK_15
150.             ;;
151.
152.         [Ss]* )
153.             chmod 666 $SDK_15
154.             ;;
155.
156.         * )
157.             echo "Wrong value: installaton aborted."
158.             exit 1
159.             ;;
160.
161.     esac
162. fi
163.
164. if [ -f $SDK_21 ]
165. then
166.
167.     echo -e $DIV
168.
169.     read -p "Installer found \"$SDK_21\" file which might defect rest of
170.     the installation process, remove?
171.
172.     [Y]es, [N]o, [S]kip :\" response
173.
174.     case $response in
175.
176.         [Yy]* )
177.             rm -f $SDK_21 2>/dev/null
178.             ;;
179.
180.         [Nn]* )
181.             chmod 666 $SDK_21
182.             ;;
183.
184.         [Ss]* )
185.             chmod 666 $SDK_21
186.             ;;
187.
188.         * )
189.             echo "Wrong value: installaton aborted."
```

```

190.         exit 1
191.         ;;
192.
193.     esac
194. fi
195.
196. if [ -f $ADT_RM ]
197. then
198.
199.     echo -e $DIV
200.
201.     read -p "Installer found \"$ADT_RM\" file which might defect rest of
202.     the installation process, remove?
203.
204.     [Y]es, [N]o, [S]kip :\" response
205.
206.     case $response in
207.
208.         [Yy]* )
209.             rm -f $ADT_RM 2>/dev/null
210.             ;;
211.
212.         [Nn]* )
213.             chmod 666 $ADT_RM
214.             ;;
215.
216.         [Ss]* )
217.             chmod 666 $ADT_RM
218.             ;;
219.
220.         * )
221.             echo "Wrong value: installaton aborted."
222.             exit 1
223.             ;;
224.
225.     esac
226. fi
227.
228.     # Is there a /opt/android-sdk-linux directory?
229.
230.     DIRECTORY=/opt/android-sdk-linux
231.
232.     if [ -d "$DIRECTORY" ]
233.     then
234.
235.         echo -e $DIV
236.
237.         read -p "Directory \"$DIRECTORY\" already exists.

```

```
238.
239.     Installer will overwrite this directory with new installation data, backup will
240.     be stored in \"$DIRECTORY.sdk.bak\", continue?
241.
242.     [Y]es, [N]o, [S]kip :\" response
243.
244.     case $response in
245.
246.     [Yy]* )
247.         # backup and/or remove existing /opt/android-sdk-linux directory
248.         cp -R $DIRECTORY $DIRECTORY.sdk.bak
249.         rm -Rf $DIRECTORY 2>/dev/null
250.         ;;
251.
252.     [Nn]* )
253.         echo -e \"\nCan't continue without access to \"$DIRECTORY\" directory\"
254.         exit 1
255.         ;;
256.
257.     [Ss]* ) ;;
258.
259.     * )
260.         echo \"Wrong value: installaton aborted.\"
261.         exit 1
262.         ;;
263.     esac
264.
265.     else
266.         mkdir $DIRECTORY
267.         chmod 755 -R $DIRECTORY
268.     fi
269.
270.     echo -e $DIV
271.
272.     # Check if SDK/Tools directory is empty
273.
274.     TOOL=\"/opt/android-sdk-linux/tools\"
275.
276.     STARTER=\"Android SDK Starter Package\"
277.
278.     if [ -z \"ls $TOOL\" ]
279.     then
280.         echo -e \"\n\nAndroid SDK Tools seem to be present, moving forward\n\"
281.     else
282.
283.         read -p \"Install $STARTER?\"
284.
285.         Provides tools such as \"android\" (Android SDK and AVD Manager),
```

```

286.     \emulator\ (Android Emulator), \ddms\ (Dalvik Debug Monitor), et cetera ...
287.
288.     [Y]es, [N]o, [S]kip: " response
289.
290.     case $response in
291.
292.     [Yy]* )
293.         echo -e "\nWill now download the $STARTER\n"
294.         wget http://dl.google.com/android/android-sdk_r18-linux.tgz
295.         tar -xzvf android-sdk_r18-linux.tgz -C /opt/
296.         echo -e "\n$STARTER successfully installed"
297.     ;;
298.
299.     [Nn]* )
300.         echo -e "\nInstallation aborted:
301.         \nCan't continue without Android Tools!\n"
302.         exit 1
303.     ;;
304.
305.     [Ss]* )
306.         echo -e "\nReluctantly skipping over this step\n"
307.     ;;
308.
309.     * ) echo "Wrong value: installaton aborted."
310.         exit 1
311.     ;;
312.
313.     esac
314.     fi
315.
316.     echo -e $DIV
317.
318.     # Install latest Android API?
319.
320.     LATEST_REV="4.0.3 API 15, revision 3"
321.     LATEST_REV_URL=http://dl.google.com/android/repository/android-15_r03.zip
322.
323.     read -p "Install latest Android API $LATEST_REV?
324.     (supports all the latest features)
325.
326.     [Y]es, [N]o, [S]kip : " response
327.
328.     case $response in
329.
330.     [Yy]* )
331.         # Download latest Android SDK
332.         echo -e "\nWill now download $LATEST_REV\n"
333.         wget $LATEST_REV_URL

```

```

334.         unzip android-15_r03.zip -d /opt/android-sdk-linux/platforms
335.         echo -e "\nSuccessfully installed Android API $LATEST_REV"
336.     ;;
337.
338. [Nn]* )
339.     echo -e "Android $LATEST_REV did NOT install\n"
340.     ;;
341.
342. [Ss]* ) ;;
343.
344. * ) echo "Wrong value: installaton aborted."
345.     exit 1
346.     ;;
347.
348. esac
349.
350. echo -e $DIV
351.
352. CLASSIC_REV="2.1, revision 03"
353. CLASSIC_REV_URL=http://dl.google.com/android/repository/android-2.1_r03-linux.zip
354.
355. read -p "Install Android API $CLASSIC_REV? (supported by ~97% phones and tablets)
356.
357. [Y]es, [N]o, [S]kip: " response
358.
359. case $response in
360.
361. [Yy]* )
362.     # Download Android 2.1 SDK
363.     echo -e "\nWill now download $CLASSIC_REV\n"
364.     wget $CLASSIC_REV_URL
365.     # ako nema android-sdk-linux/platforms moras napraviti?
366.     unzip android-2.1_r03-linux.zip -d /opt/android-sdk-linux/platforms
367.     echo -e "\nSuccessfully installed Android API $CLASSIC_VERSION"
368.     ;;
369.
370. [Nn]* )
371.     echo -e "Android $CLASSIC_REV did NOT install\n"
372.     ;;
373.
374. [Ss]* ) ;;
375.
376. * ) echo "Wrong value: installaton aborted."
377.     exit 1
378.     ;;
379.
380. esac
381.

```

```

382.     echo -e $DIV
383.
384.     # Configure PATH environmental variable
385.
386.     PATHV=/etc/profile.d/android-sdk.sh
387.
388.     read -p "Configure PATH environmental variable?"
389.
390.     Will allow you to run tools such as \"android\" (Android SDK and AVD Manager),
391.     \"emulator\" (Android Emulator), \"ddms\" (Dalvik Debug Monitor) directly from
392.     Terminal.
393.
394.     [Y]es, [N]o, [S]kip: " response
395.
396.     case $response in
397.
398.     [Yy]* )
399.         if [ -f $PATHV ]
400.         then
401.
402.             read -p "
403.             An existing \"$PATHV\" file has been detected in \"/etc/profile.d/"
404.
405.             Installer will overwrite this file, backup will be made and stored in
406.             \"$PATHV.sdk.bak\"
407.
408.             Continue?
409.
410.             [Y]es, [N]o, [S]kip : " response
411.
412.             case $response in
413.
414.             [Yy]* )
415.                 # Backup and remove current android-sdk.sh file
416.                 cp -f $PATHV $PATHV.sdk.bak
417.                 rm -f $PATHV 2>/dev/null
418.             ;;
419.
420.             [Nn]* )
421.                 echo -e "\PATH environmental variable was NOT configured, Goodbye"
422.                 exit 1
423.             ;;
424.
425.             [Ss]* ) ;;
426.
427.             * )
428.                 echo "Wrong value: installaton aborted."
429.                 exit 1

```

```
430.     ;;
431.
432.     esac
433.
434.     else
435.
436.         touch $PATHV
437.         chmod 644 $PATHV
438.
439.     fi
440.
441.     # Create android-sdk.sh file
442.
443.     echo -n '#!/bin/bash -e
444.     # PATH for Android SDK
445.     PATH="/opt/android-sdk-linux/tools:/opt/android-sdk-
linux/platforms:$PATH"' > $PATHV
446.
447.         # apply PATH without user having to log out
448.         export PATH="/opt/android-sdk-linux/tools:/opt/android-sdk-
linux/platforms:$PATH"
449.
450.         # make sure permissions are alright
451.         chmod -R 755 $DIRECTORY
452.
453.         # in future add "sed" here in order to avoid repetitive code
454.
455.         echo -e "\nPATH successfully added and configured."
456.     ;;
457.
458.     [Nn]* )
459.         echo -e "\nWon't continue without configuring PATH environmental variable"
460.         exit 1
461.     ;;
462.
463.     [Ss]* ) ;;
464.
465.     * )
466.         echo "Wrong value: installaton aborted."
467.         exit 1
468.     ;;
469.
470. esac
471.
472. echo -e $DIV
473.
474. # Eclipse and Android ADT plugin installation
475.
```

```
476.     read -p "Install ADT (Android Development Tools) Plugin for Eclipse IDE?"
477.
478.     [Y]es, [N]o, [S]kip: " response
479.
480.     case $response in
481.
482.     [Yy]* )
483.         PACKAGE=Eclipse
484.
485.         if dpkg -s "$PACKAGE" 2>/dev/null 1>/dev/null
486.         then
487.             echo -e "\n$PACKAGE is installed.\nWill now download Eclipse ADT Plugin\n"
488.
489.             wget http://dl.google.com/android/ADT-18.0.0.zip
490.             unzip ADT-18.0.0.zip -d /tmp/
491.
492.             cp /tmp/features/*.jar /usr/share/eclipse/features/
493.             cp /tmp/plugins/*.jar /usr/share/eclipse/plugins/
494.             # see what to do with rest of the files web/ index.html & site.xml
495.
496.         else
497.
498.             read -p "
499. $PACKAGE IDE is NOT installed, would you like to install it?
500. [Y]es, [N]o, [S]kip : " response
501.
502.             case $response in
503.
504.             [Yy]* )
505.                 apt-get install eclipse
506.
507.                 echo -e "\nOkay, will now download Eclipse ADT Plugin\n"
508.
509.                 wget http://dl.google.com/android/ADT-18.0.0.zip
510.                 unzip ADT-18.0.0.zip -d /tmp/
511.
512.                 cp /tmp/features/*.jar /usr/share/eclipse/features/
513.                 cp /tmp/plugins/*.jar /usr/share/eclipse/plugins/
514.                 # see what to do with rest of the files web/ index.html & site.xml
515.
516.             ;;
517.
518.             [Nn]* )
519.                 echo -e "\nADT Plugin can't be installed without $PACKAGE IDE"
520.                 exit 1
521.             ;;
522.
523.             [Ss]* ) ;;
```

```
524.         * )
525.         echo "Wrong value: installaton aborted."
526.         exit 1
527.         ;;
528.
529.         esac
530.     fi
531. esac
532.
533. echo -e $DIV
534.
535. # Adds Hardware Device support by adding udev rules
536.
537. RULES=/etc/udev/rules.d/51-android.rules
538.
539. read -p "Add hardware support for you Android devices?"
540.
541. [Y]es, [N]o, [S]kip : " response
542.
543. case $response in
544.
545.     [Yy]* )
546.         echo "Adding hardware support"
547.         ;;
548.
549.     [Nn]* )
550.         echo "Hardware support was NOT added"
551.         ;;
552.
553.     [Ss]* ) ;;
554.
555.     * )
556.         echo "Wrong value: installaton aborted."
557.         exit 1
558.         ;;
559.
560.     esac
561.
562. # Check if the "rules" file already exists
563. if [ -f $RULES ];
564. then
565.
566.     read -p "
567.     An existing \"51-android.rules\" file has been detected in \"/etc/udev/rules.d/"
568.
569.     Installer will overwrite this file, backup will be made and stored in
570.     \"$RULES.sdk.bak\"
571.
```

```

572.         Continue?
573.
574.         [Y]es, [N]o, [S]kip : " response
575.
576.         case $response in
577.
578.             [Yy]* )
579.                 # backup and remove current 51-android.rules file
580.                 cp -f $RULES $RULES.sdk.bak
581.                 rm -f $RULES 2>/dev/null
582.                 ;;
583.
584.             [Nn]* ) echo -e "\nInstaller did NOT add hardware support, Goodbye"
585.                 exit 1
586.                 ;;
587.
588.             [Ss]* ) ;;
589.
590.             * ) echo "Wrong value: installaton aborted."
591.                 exit 1
592.                 ;;
593.
594.         esac
595.
596.         else
597.
598.             # create rules files and give it proper permissions
599.             touch $RULES
600.             chmod 644 $RULES
601.
602.         fi
603.
604.         # add new 51-android.rules file
605.         echo -n '# udev rules which add hardware device support on Linux
606.         #
607.         # "Using Hardware devices"
608.         # (http://developer.android.com/guide/developing/device.html)
609.         #
610.         # List last updated: May 9, 2012
611.
612.         # Acer
613.         SUBSYSTEM=="usb", ATTR{idVendor}=="0502", MODE="0666", OWNER="plugdev"
614.
615.         # ASUS
616.         SUBSYSTEM=="usb", ATTR{idVendor}=="0b05", MODE="0666", OWNER="plugdev"
617.
618.         # Dell
619.         SUBSYSTEM=="usb", ATTR{idVendor}=="413c", MODE="0666", OWNER="plugdev"

```

```
620.
621.     # Foxconn
622.     SUBSYSTEM=="usb", ATTR{idVendor}=="0489", MODE="0666", OWNER="plugdev"
623.
624.     # Fujitsu
625.     SUBSYSTEM=="usb", ATTR{idVendor}=="04c5", MODE="0666", OWNER="plugdev"
626.
627.     # Fujitsu Toshiba
628.     SUBSYSTEM=="usb", ATTR{idVendor}=="04c5", MODE="0666", OWNER="plugdev"
629.
630.     # Garmin-Asus
631.     SUBSYSTEM=="usb", ATTR{idVendor}=="091e", MODE="0666", OWNER="plugdev"
632.
633.     # Google
634.     SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666", OWNER="plugdev"
635.
636.     # Hisense
637.     SUBSYSTEM=="usb", ATTR{idVendor}=="109b", MODE="0666", OWNER="plugdev"
638.
639.     # HTC
640.     SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", OWNER="plugdev"
641.
642.     # Huawei
643.     SUBSYSTEM=="usb", ATTR{idVendor}=="12d1", MODE="0666", OWNER="plugdev"
644.
645.     # K-Touch
646.     SUBSYSTEM=="usb", ATTR{idVendor}=="24e3", MODE="0666", OWNER="plugdev"
647.
648.     # KT Tech
649.     SUBSYSTEM=="usb", ATTR{idVendor}=="2116", MODE="0666", OWNER="plugdev"
650.
651.     # Kyocera
652.     SUBSYSTEM=="usb", ATTR{idVendor}=="0482", MODE="0666", OWNER="plugdev"
653.
654.     # Lenovo
655.     SUBSYSTEM=="usb", ATTR{idVendor}=="17ef", MODE="0666", OWNER="plugdev"
656.
657.     # LG
658.     SUBSYSTEM=="usb", ATTR{idVendor}=="1004", MODE="0666", OWNER="plugdev"
659.
660.     # Motorola
661.     SUBSYSTEM=="usb", ATTR{idVendor}=="22b8", MODE="0666", OWNER="plugdev"
662.
663.     # NEC
664.     SUBSYSTEM=="usb", ATTR{idVendor}=="0409", MODE="0666", OWNER="plugdev"
665.
666.     # Nook
667.     SUBSYSTEM=="usb", ATTR{idVendor}=="2080", MODE="0666", OWNER="plugdev"
```

```
668.
669.     # Nvidia
670.     SUBSYSTEM=="usb", ATTR{idVendor}=="0955", MODE="0666", OWNER="plugdev"
671.
672.     # OTGV
673.     SUBSYSTEM=="usb", ATTR{idVendor}=="2257", MODE="0666", OWNER="plugdev"
674.
675.     # Pantech
676.     SUBSYSTEM=="usb", ATTR{idVendor}=="10a9", MODE="0666", OWNER="plugdev"
677.
678.     # Pegatron
679.     SUBSYSTEM=="usb", ATTR{idVendor}=="1d4d", MODE="0666", OWNER="plugdev"
680.
681.     # Philips
682.     SUBSYSTEM=="usb", ATTR{idVendor}=="0471", MODE="0666", OWNER="plugdev"
683.
684.     # PMC-Sierra
685.     SUBSYSTEM=="usb", ATTR{idVendor}=="04da", MODE="0666", OWNER="plugdev"
686.
687.     # Qualcomm
688.     SUBSYSTEM=="usb", ATTR{idVendor}=="05c6", MODE="0666", OWNER="plugdev"
689.
690.     # SK Telesys
691.     SUBSYSTEM=="usb", ATTR{idVendor}=="1f53", MODE="0666", OWNER="plugdev"
692.
693.     # Samsung
694.     SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", MODE="0666", OWNER="plugdev"
695.
696.     # Sharp
697.     SUBSYSTEM=="usb", ATTR{idVendor}=="04dd", MODE="0666", OWNER="plugdev"
698.
699.     # Sony
700.     SUBSYSTEM=="usb", ATTR{idVendor}=="054c", MODE="0666", OWNER="plugdev"
701.
702.     # Sony Ericsson
703.     SUBSYSTEM=="usb", ATTR{idVendor}=="0fce", MODE="0666", OWNER="plugdev"
704.
705.     # Teleepoch
706.     SUBSYSTEM=="usb", ATTR{idVendor}=="2340", MODE="0666", OWNER="plugdev"
707.
708.     # Toshiba
709.     SUBSYSTEM=="usb", ATTR{idVendor}=="0930", MODE="0666", OWNER="plugdev"
710.
711.     # ZTE
712.     SUBSYSTEM=="usb", ATTR{idVendor}=="19d2", MODE="0666", OWNER="plugdev"
713.     ' > $RULES
714.
715.     # in future add "sed" in order to avoid repetitive code
```

```

716.
717.     # provide all groups, users and others with read permissions for this file
718.     chmod a+r $RULES
719.
720.     echo -e "\nHardware device support successfully installed"
721.
722.     echo -e $DIV
723.
724.     # add MTP support
725.
726.     read -p "Add MTP support? (enables USB mass storage support on Android >= 4.0)
727.
728.     [Y]es, [N]o, [S]kip: " response
729.
730.     case $response in
731.
732.     [Yy]* )
733.         PACKAGE=mtpfs
734.
735.         if [ dpkg -s "$PACKAGE" 2>/dev/null 1>/dev/null ]
736.             then
737.                 echo -e "\nIt seems you have all the needed packages installed"
738.
739.             else
740.
741.                 read -p "
742. Packages seem to be missing, install required packages?
743. [Y]es, [N]o : " response
744.
745.                 case $response in
746.
747.                     [Yy]* )
748.                         apt-get install mtp-tools mtpfs
749.                         ;;
750.
751.                     [Nn]* )
752.                         echo "Did not install MTP support"
753.                         # exit 1
754.                         ;;
755.
756.                     * )
757.                         echo "Wrong value: installaton aborted."
758.                         exit 1
759.                         ;;
760.
761.                 esac
762.
763.                 echo -e "\nMTP support successfully installed"

```

```

763.
764.         # in future add device name to /media dir
765.         # fix permissions and add user to fuse group
766.         # edit fuse.conf and .bashrc
767.         fi
768.     ;;
769.
770.     [Nn]* )
771.         echo "Did NOT enable MTP support"
772.         exit 1
773.         ;;
774.
775.     [Ss]* ) ;;
776.
777.     * )
778.         echo "Wrong value: installation aborted."
779.         exit 1
780.         ;;
781.
782. esac
783.
784.     # clean up
785.
786.     SDK_R18="android-sdk_r18-linux.tgz"
787.     SDK_15="android-15_r03.zip"
788.     SDK_21="android-2.1_r03-linux.zip"
789.     ADT_RM="ADT-18.0.0.zip"
790.
791.     if [ -f $SDK_R18 ]
792.     then
793.
794.         echo -e $DIV
795.
796.         read -p "Installer left \"${SDK_R18}\" file, would you like to remove it?"
797.
798.         [Y]es, [N]o, [S]kip : " response
799.
800.     case $response in
801.
802.         [Yy]* )
803.             rm -f $SDK_R18 2>/dev/null
804.             ;;
805.
806.         [Nn]* )
807.             chmod 666 $SDK_R18
808.             ;;
809.
810.         [Ss]* )

```

```
811.         chmod 666 $SDK_R18
812.         ;;
813.
814.         * )
815.         echo "Wrong value: installaton aborted."
816.         exit 1
817.         ;;
818.
819.     esac
820. fi
821.
822. if [ -f $SDK_15 ]
823. then
824.
825.     echo -e $DIV
826.
827.     read -p "Installer found \"$SDK_15\" file which might defect rest of
828.     the installation process, remove?
829.
830.     [Y]es, [N]o, [S]kip :\" response
831.
832.     case $response in
833.
834.         [Yy]* )
835.             rm -f $SDK_15 2>/dev/null
836.             ;;
837.
838.         [Nn]* )
839.             chmod 666 $SDK_15
840.             ;;
841.
842.         [Ss]* )
843.             chmod 666 $SDK_15
844.             ;;
845.
846.         * )
847.             echo "Wrong value: installaton aborted."
848.             exit 1
849.             ;;
850.
851.     esac
852. fi
853.
854. if [ -f $SDK_21 ]
855. then
856.
857.     echo -e $DIV
858.
```

```
859.     read -p "Installer left \"$SDK_21\" file, would you like to remove it?"
860.
861.     [Y]es, [N]o, [S]kip : " response
862.
863.     case $response in
864.
865.         [Yy]* )
866.             rm -f $SDK_21 2>/dev/null
867.             ;;
868.
869.         [Nn]* )
870.             chmod 666 $SDK_21
871.             ;;
872.
873.         [Ss]* )
874.             chmod 666 $SDK_21
875.             ;;
876.
877.         * )
878.             echo "Wrong value: installaton aborted."
879.             exit 1
880.             ;;
881.
882.     esac
883. fi
884.
885. if [ -f $ADT_RM ]
886. then
887.
888.     echo -e $DIV
889.
890.     read -p "Installer left \"$ADT_RM\" file, would you like to remove it?"
891.
892.     [Y]es, [N]o, [S]kip : " response
893.
894.     case $response in
895.
896.         [Yy]* )
897.             rm -f $ADT_RM 2>/dev/null
898.             ;;
899.
900.         [Nn]* )
901.             chmod 666 $ADT_RM
902.             ;;
903.
904.         [Ss]* )
905.             chmod 666 $ADT_RM
906.             ;;
```

```
907.  
908.         * )  
909.         echo "Wrong value: installaton aborted."  
910.         exit 1  
911.         ;;  
912.  
913.     esac  
914. fi  
915.  
916. echo -e $DIV  
917.  
918. echo -e "Installation complete. Feel free to run \"android-sdk-installer\" at any  
919. time to (re)configure.\n\nEnjoy!\n";  
920.  
921. exit 0
```