

Using Android in Education for Mobile Device Development

Chris Baker and Danny Noler
NSF-REU 2008

July 2, 2008

Abstract

In the growing world of technology, cellular devices have quickly emerged as one of the fastest evolving fields. They have increased greatly in both popularity and complexity, requiring more advanced operating systems and applications to meet the demands of the consumer. Android is a software stack designed to meet these demands in an open source environment. The project is currently being developed and funded by the Open Handset Alliance, which includes companies such as Google and T-Mobile. Android includes an operating system, middleware, and key applications, as well as a Software Development Kit (SDK) for developers to create their own applications for the Android environment. Due to its open source license and tools provided, Android is an ideal platform for bringing the mobile market to the educational realm.

This paper describes in detail what Android is, its architecture, and why developers should choose Android to develop in a mobile device platform environment. The paper also discusses the basic hardware requirements for porting the current version of Android to real hardware. Finally, it discusses some of the obstacles we are faced with, our achievements with porting Android to a Nokia N800 and a Sharp Zaurus SL-C3200, and the lessons we learned during this process.

1 What is Android?

Android is a software stack currently being developed by Google for mobile devices. In 2007 the Open Handset Alliance was formed, which consists of more than 30 technology and mobile companies such as Intel, HTC, Motorola, etc. This alliance is headed by Google, with each company helping to develop open standards for mobile devices. Android is the first complete and open mobile platform using this new standard. Android uses a Linux kernel (version 2.6) to achieve many core system services such as memory management, security, and even a network stack to provide a secure and stable environment for its architecture. The overlying software consists of three parts that work together to form a working software stack for mobile devices. Android includes its own userspace on top of the Linux kernel, which contains the APIs necessary for Android applications. It is also responsible coordinating activities that take place within the computer. Above the operating system lies middleware components. This is the software that connects software components or other applications to each other. Middleware is used to distribute services across multiple processes running on the computer. Key applications combine to form the final part of Android. This includes the basic applications found on most smart phones that are on the market today. A browser, contact list, map application, and the phone software itself are all key applications that are found within the Android architecture.

2 Why Choose Android?

In order to attract developers, Google promotes four main features of Android. The first is its open nature. This means that Android's API will be freely available to any programmer who wishes to create an Android application. The second feature is the application hierarchy. All applications are on an equal playing field, meaning even a phone's basic applications (such as the dialer) can be replaced by third party programs. Android also combines information from the web with data on the phone. For example, one could use contact information stored on the phone to determine the geographic location of the contact's home using information stored online. Finally, Android includes a Software Development Kit in order to expediate application design and implementation. The SDK includes everything a developer needs to begin writing Android applications, which are written in the Java programming

language.

3 Hardware Requirements for Android

Although the source code for Android has not been officially released, it is still possible to port Android to real hardware. By using a compiled Linux kernel and applying the necessary Android patches, one can port Android to a real device and run a version of Android on the device. Android is currently very hardware specific and when choosing a device to port to, one must abide by the minimum hardware requirements. Since Google has not released the source code to Android, only code available in the SDK can be used. The SDK itself uses an open source processor emulator, known as QEMU, to create a virtual ARM system-on-a-chip (SoC) known as Goldfish. The ARM SoC boots a 2.6.23 ARM Linux kernel with support for the Goldfish platform on a x86 host. Since Goldfish contains software compiled for the ARMv5 instruction set, only hardware with an ARM926 or higher can be used. Sufficient main memory is 32MB, but this is only enough to run Android at a very basic level and should only be used as a proof of concept. A higher amount of main memory should be used in order to obtain a usable system. These are the minimum hardware prerequisites that must be met in order to port and run Android on real hardware. There have been many documented successful attempts at porting Android to real hardware. This paper also shares some of our successes and achievements with porting Android to a device. [5]

4 Educational Use

A great deal of work has been done with Android, but one might wonder what purpose work like ours serves. Our ultimate goal to show that Android is an excellent tool for learning. Given its open source nature, there is a great deal of resources available for both programmers and hackers. Google has already released a full SDK for Android, allowing developers to write applications and test them using an emulator, although applications can be ported to devices on which Android has been hacked and test them there. There is also a thriving community of enthusiasts who provide insights into Android's mechanics and their experiences with porting the software. This provides an

ideal environment for educating others on mobile device development, which has a different philosophy from desktop development (emphasis on efficiency and use of memory is critical in order to save battery and prevent system lockup, for example). Also, hacking Android onto different mobile devices provides some insight into cross-compiling and device driver programming. Given the current trend of continually shrinking computers and electronic devices, it is vital that the next generation of computer scientists and engineers have a grasp on mobile development. Android gives these students an opportunity to see the front lines of development. Once Android is finally completed and released on actual devices, its community is destined to grow further, and application development will become more widespread. In turn, more resources will become available, and students will find an information gold mine on the subject. Also, the release of the source code will allow upcoming engineers and computer scientists to see the physical code that goes into a mobile platform.

5 Obstacles

There are a number of obstacles that must be overcome in order to work with Android. The greatest and most obvious is the learning curve. Those familiar with Linux kernels, device driver programming, and cross compilers will no doubt have an advantage while porting Android. Also, the architecture must be understood before programming any application to run on an Android device. Both of these presented an initial (and continuing) challenge to us, as neither of us have ever worked this deeply with Linux. This is also our first work with Android. However, these issues can be overcome through literature and other sources of information. Google provides an impressive guide for Android and application programming. Message boards are also available for communication with other developers. Linux development also has a wide selection of resources both online and in print. It is up to each individual to determine which sources best meet their needs.

There is one other large obstacle, though not permanent. Since Google has not released an official Android device for purchase, they have not released the full source code to Android. This is the reason why an ARM architecture is required for porting Android, as the source has not been compiled for x86 processors. Therefore, Android must be "patched" together using a combination of a Linux kernel, Android patches specific to a particu-

lar board, and any other patches needed for hardware fixes such as keyboard and touchscreen functionality. While this does allow for an added difficulty that some might find exciting and challenging, it also prevents Android from being used on machines that would otherwise meet basic Android requirements. The first Android device is expected in Q4 2008 or Q1 2009, at which point the source code for Android will be released to the public. At this time, this obstacle can only be overcome through ingenuity and through the successes of others who post their stories online.

6 Achievements and Lessons Learned

6.1 Android on Nokia N800

The first real hardware device that we began work with was the Nokia N800 internet tablet. This device has WiFi and touchscreen capabilities which Android supports. One of the first steps we took with the N800 was to update the native operating system to OS2008. Before beginning the process of porting Android to the N800, we downloaded and installed the necessary tools and files that would be needed to achieve a successful port of the Android image. These files included the 2.6.21 Linux kernel, an ARM GNU Toolchain for cross-compiling, Maemo flasher 3.0, prebuilt Android file system, touchscreen patch specific to the N800, and an Android patch which was derived from the SDK and is specific for the N800 (version 1). There are newer versions of this patch (versions 2 and 3) available, but these patches were designed specifically for the N810 and will not be fully compatible for the N800. A microSD card was formatted using the ext2 standard and the system files were then copied to the card.[4]

We began the actual process of porting Android to the N800 by applying the necessary patches to the Linux kernel source. First we patched the kernel source with the Android version 1 patch followed by applying the N800 touchscreen patch to the source. After the patches were applied, we cross-compiled the source using the ARM GNU Toolchain in order to obtain a version that was compatible with an ARM architecture. An Android image file is produced by this cross-compilation. The Maemo flasher 3.0 tool was then used to flash the image to the N800 device. Once the image was successfully flashed to the N800, we took the microSD card containing the Android file system and placed it in the N800. By using a custom startup



Figure 1: Chris holding the N800 with Android running

script designed to mount the file system on the N800, we were then able to successfully start Android and obtain a working Android device. Please note the native OS2008 is not overwritten, and is still the default system booted when the N800 is turned on. The startup script must be run from the Terminal once inside OS2008.[9]

It should be mentioned that we used the m3 release of Android for the user space while using the m5 release to patch the Linux kernel. This is done because the Linux kernel will not support some of the newer features on the m5 system, such as double buffer support in the frame buffer driver. Also, the N800 lacks both a physical and virtual keyboard. Therefore, it is not possible to test text input with the web browser once inside of Android.

6.2 Android on Sharp Zaurus SL-C3200

Our final device was the Zaurus SL-C3200. This device has several advantages over the Nokia N800, including a full QWERTY keyboard, a six gigabyte hard drive, and a faster processor. However, it proved to be more challenging than the N800. Initially, we just wanted to check Android's usability on the device. We found an Android enthusiast who had uploaded pre-compiled images of the Linux kernel and Android m3 userspace for the C3200. We downloaded the files, copied them to an SD card, and updated the Zaurus using its built in update tool. All went well, and within ten minutes we had a working Android device. Unfortunately, the touch screen was not functional, and without a built-in WiFi card, we also could not access the Internet. However, all the applications were functional.[2]

From there, we decided to attempt to compile our own linux kernel. With this method, we could apply touch screen patches and try the m5 userspace, which some developers had gotten to work on the C3200. Before we started with the Zaurus, we first had to download and install OpenEmbedded, an open source cross-compiler. This was a hassle in itself, as there was not much documentation on configuring and building the tool chain. After several days of trial and error, we were finally able to build the tool using instructions spliced from several websites.[3, 6]

After building the tool, we compiled a linux kernel image for the C3200 architecture. This particular image is known as the Angstrom Distribution, and was created specifically for embedded devices. We altered some configuration files in order to apply the necessary Android patches to the kernel. The compilation itself took almost half a day. From there, we copied the Linux files over to an SD card and updated the Zaurus in the same fashion as before. Once the system was successfully formatted, we had a working Linux distribution, though there was no GUI as of yet. From here, we extracted the Android file system. Luckily, Google provides these files as an image file in their SDK. We extracted the image on the Zaurus, created a small startup script, and fired up Android. Unfortunately, Android would not load past its moving red cylon bar. After doing some research online, we discovered we were missing a kernel module called Binder. We managed to find Binder online, and attempted to insert it into the kernel. However, the Binder module was created for Linux kernel version 2.6.23, and 2.6.26 was installed on the Zaurus. At this point we were near the end of our summer session, and unfortunately could not determine how to build an Angstrom



Figure 2: Danny holding the C3200 with Android running

Distribution with kernel version 2.6.23. However, this project is not without its successes. We learned a great deal about cross compilers, linux make files, and scripts.[3, 8]

7 Conclusion

Android has come a long way since its inception three years ago. It will surely impact the mobile market when the first Android phones are released, but also provides an excellent opportunity for developers to enter the mobile market. A great deal can be learned by both creating applications using Android's unique Java API as well as attempting to port Android to real hardware. The classroom can benefit from Android, given the amount of resources available for reference. Given the market trend towards the mobile market, future developers can greatly benefit by using Android to improve

their algorithms and resource management. Since it is open source, there is no cost to learn. Android is a win-win situation for both the community and developers.

References

- [1] Alextreme. Android on n810 - binaries and a fixed tutorial. http://www.alextreme.org/drupal/?q=android_on_n810, May 2008. Blog entry where m3 Android userspace was downloaded.
- [2] Cortez. Android on zaurus. <http://www.omegamoons.com/blog/static.php?page=ZaurusAndroid>, July 2008. Blog entry where m3 Android files were downloaded for Zaurus.
- [3] Andy Dinosaur. Google android runs on sharp zaurus sl-c3000. <http://androidzaurus.seesaa.net/article/74237419.html>, December 2007. Blog entry specifying how to use OpenEmbedded to put Android on Zaurus.
- [4] User generated Wiki. Android on nokia n8xx. <http://code.google.com/p/android-on-n8xx/wiki/Home>, June 2008.
- [5] User generated Wiki. Android on omap. http://elinux.org/Android_on_OMAP, 2008.
- [6] User generated Wiki. Getting started. http://wiki.openembedded.net/index.php/Getting_Started, June 2008. How to install OpenEmbedded tool chain and compile images.
- [7] Google. Android home page. <http://code.google.com/android>, 2008.
- [8] lardman. Spitz installation instructions. <http://www.angstrom-distribution.org/angstrom-installation-spitz>, May 2007. How to install Angstrom on Spitz (SL-C3200 hardware).
- [9] Nuuneol. http://groups.google.com/group/android-internals/browse_thread/thread/7028432fa76f57e8/6d77d401e5076000, May 2008. Message board post where Nokia N800 touchscreen patch was downloaded.