TEX and friends on a Pad

Boris Veytsman

Abstract

TEX on an Eee Pad is quite workable.

1 Introduction

Some time ago a blog entry [15] made quite a splash in the community. The (semi-anonymous) author stated that LATEX cannot be made on a tablet due to its "speed, bloat, and complexity" and needs a complete rewrite. He also asked for a complete change in the licensing scheme of TEX components in order to make LATEX acceptable for the App Store.

In my opinion, this is a complete misunderstanding of what TEX is and what it is not. The most important thing, TEX is not an "app" in the same sense OpenOffice is. TEX is designed as a compiler which takes a program written in a language understandable by humans, and creates "binary code" in a language understood by machines. The `tex` files we write are not "documents" in the same sense as OpenOffice files. They are *programs* with familiar (to a programmer) constructions like macros, loops, conditionals. The result of compilation is code — a DVI, a PS or a PDF file — which is basically a set of instructions for a machine to produce printed pages or images on a screen. Furthermore, the TEX system does not have just one compiler, but a family of compilers and utilities, like `gcc` and friends. While the article [15] exclusively discusses LATEX, it is nice to have index processors, bibliography formatters, font manipulation utilities and many others, not to mention alternative engines to `pdfetex` and alternative formats to LATEX.

Once we understand that we are talking about a family of compilers with auxiliary programs and libraries, many objections in [15] become irrelevant. The compilation of the engines is a complex process with many helper applications? Anybody who ever tried to bootstrap `gcc` from source would not make this comment. Huge code base? Well, the code base of C/C++ with *all* the free libraries commonly used is not small, either.

The comparison of a TEX distribution to a C/C++ distribution including all possible libraries is not as far fetched as it seems. A modern distribution like TEX Live contains almost all the freely distributable code from CTAN, the Comprehensive TEX Archive Network. The users of Perl and R have created and maintain similar huge collections — CPAN and CRAN. It is commonly considered to be the strong feature of these languages rather than a weakness.

The minimal subset of TEX Live occupies about 40 Mb — by no means large by today's standards. The full distribution, indeed, is 3.5 Gb and growing, because it includes solutions for many different problems: typesetting musical scores and chess games, working with many languages and scripts, drawing geographic maps in any projection, creating circuit diagrams, using medieval fonts, and many, many others. A user can install only the parts which she really needs: for somebody the main reason to work with TEX might be the possibility to typeset in the Klingon language, while another user might work with TEX for years and never find out it speaks Klingon. Fortunately, modern distributions provide easy and powerful tools to select only the parts of TEX and friends one really needs. And modern hard disks are large enough that installing the full distribution is a reasonable default.

Still, the proof of the pudding is in the eating, so we would like to offer the most convincing proof that TEX can be used on a tablet: the experience of compiling and using TEX Live on one. This is the aim of this paper.

2 Device

There are reports of running TEX on Apple IOS devices [3]. However, due to the usability considerations discussed below I chose an Android tablet on `armv7l` architecture. I recently got an ASUS Transformer Eee Pad TF101 [2]. The selling point was the dual nature of the device: it has a detachable keyboard with an extra battery, so it can be used both as a light tablet (when the lower part is detached) and a netbook (when the lower part is attached). It has turned out to be a very useful and surprisingly powerful machine.

Another advantage of this choice is that Android-OS is a derivative of GNU/Linux, so I hoped to use the familiar Linux tool chain for working with it. I found out that one can actually install a full distribution as an application, running in a `chroot` environment, which made my task quite simple.

2.1 Rooting

To really own the device one need to "get a root" on it. This is a dangerous operation *which may break your device and almost certainly voids your warranty.* Please do not do this unless you absolutely understand what you are doing, and in no circumstances blame me if anything goes wrong!

The operation is described in detail in [1]. After rooting the device you need terminal access and (optionally) a convenient shell to work before you

start Linux. Android Terminal Emulator [12] and BusyBox [16] are good choices.

For Emacs and vi users it is useful to map the "back" key on the dock to Escape (see, e.g. [7]) by editing `/system/usr/keylayout/asusec.kl`.

2.2 Linux in `chroot`

The user interface of Android assumes working with one full-screen application at any time. This is a challenge for compilers like TEX and friends: one needs an editor window, a compiler window, a log window, etc. One way to solve this challenge is to use an Integrated Developed Environment like TEXworks [8] or TEXnicCenter [17], where the window management is done by the application. However, I am an incorrigible Emacs user with the fingers used to all those Control- and Meta-sequences, so it would make most sense to recreate the familiar work flow on Android. Emacs is well integrated in Unix-like systems, and I looked for the way to install a Linux environment on the device. There are several ways to do so:

1. Dual boot. The system can be booted to Linux *or* to Android. At any time the device is either Linux or Android, without much integration between these two.

2. Virtual machine. The host operating system (Android) emulates the hardware for the guest operating system (Linux). There is some integration between the OSes, but this requires a lot of processing power. Besides, we are not aware of any VM software with Android as host (while there are many VMs with Android as guest).

3. Chroot. This is a unique possibility due to the fact that Android uses the Linux kernel with the same system calls. Thus, one can just start the standard Linux daemons and programs under Android with a separate directory mimicking the standard Linux layout (the "root directory" for these processes, hence the name).

I chose that last possibility. It should be stressed that the Linux programs with this solution are tightly integrated with the native Android system. In Figure 1, the `top` program shows both Android processes (e.g. `ys.android.jump`) and Linux processes (e.g. `top` itself).

The application "Linux Installer" [13] turned out to be an excellent way to go. I used it to install Debian Squeeze on a 32 Gb removable SD card.

To allow a non-root user to run useful processes, the user must be granted some rights to the corresponding devices on the tablet. This is done by adding the user to the groups listed in Table 1 in the file `/etc/group` on the Linux side.[1]

Group	gid
`AID_NET_BT_ADMIN`	3001
`AID_NET_BT`	3002
`AID_INET`	3003
`AID_NET_RAW`	3004
`AID_NET_ADMIN`	3005
`AID_MISC`	9998
`AID_SDCARD`	1015

Table 1: Some useful Android groups

To get X running on the Android I used the following trick, taken from an Android forum [5]. One can start a "headless" X accessed from a remote computer through the VNC protocol. The interesting thing is that this "remote" computer can actually be the *local* machine with a VNC client talking to the server on the same device at IP address 127.0.0.1. There are VNC clients for Android, normally used to remotely access computers from a mobile device. That one can deploy them to access *the same device* is a good example of the power of unintended use.

On the server side I installed TightVNC [6], included in the Debian distribution. There are a number of free VNC clients for Android. Unfortunately I did not find a single one that provided easy access to Escape and Control keys, which are essential for Emacs users. Therefore I chose Jump VNC [14]; this is the only non-free software used in this project. Jump VNC understands Control and Escape keys on the keyboard dock and provides a convenient on-screen panel with special keys when the dock is disconnected. I hope some free VNC client can implement this convenient interface in the future.

I did not use resource heavy environments like KDE or Gnome; instead, I installed a lightweight window manager, FVWM.

Some additional screenshots of the resulting desktop can be found at `http://android.galoula.com/screenshots/LinuxInstall/Boris_Veytsman_2011_12_31/`.

3 Installing and running TEX

Debian Squeeze has TEX in the distribution, so it runs "out of the box". Unfortunately, it is the very old TEX Live 2009.[2] As a nice exercise, I decided to build TEX Live 2011 binaries from sources — maybe

[1] I am grateful to Gaël Person for this advice.

[2] Debian is famous for its stability, which means, among other things, rather obsolete packages.

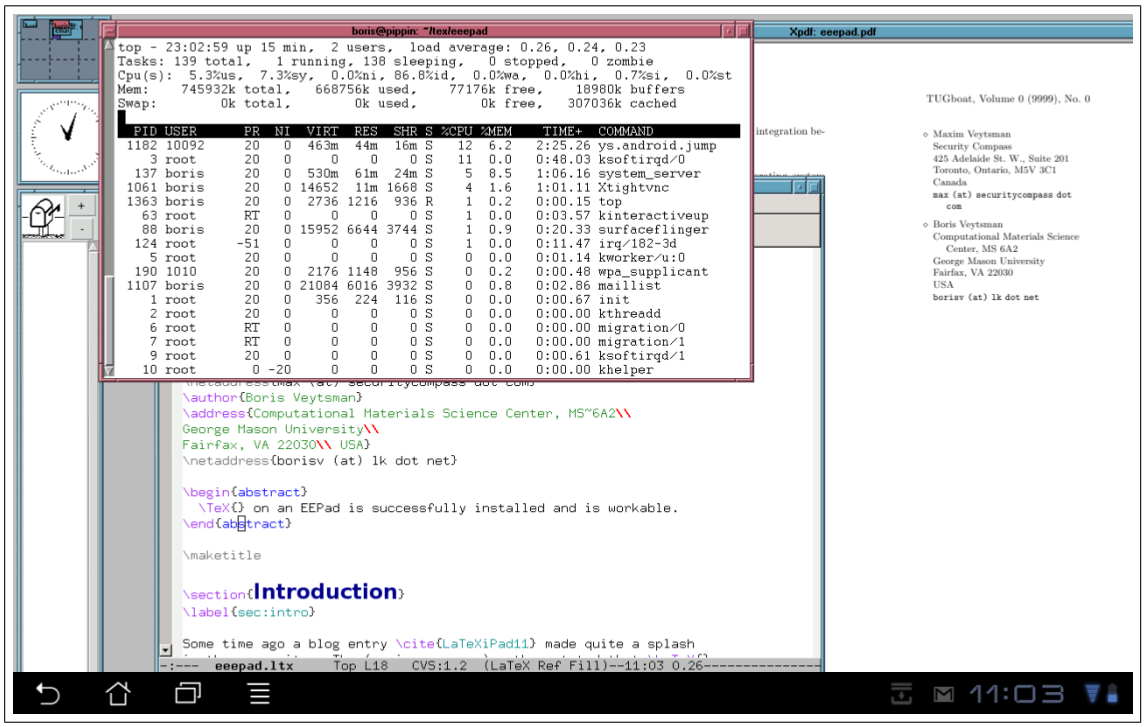

Figure 1: A screenshot with `top` running

the statement in [15] that the author could not do this was an additional incentive.

The build instructions on the TEX Live web page [10] were easy to follow. The full build took about 2.5 hours. I decided to install all TEX Live packages: being a TEX consultant, I prefer the full installation since one never knows what a next customer might need. The installation of the binaries and packages went without a problem.

The subsequent TEX Live 2012 builds on this machine also proceeded without errors, compiling all 350 binaries. This `armel-linux` port became an official part of TEX Live with the 2012 release.

The update cycle of LuaTEX and ConTEXt is typically faster than that of TEX Live. I also maintain a ConTEXt standalone distribution (see `http://wiki.contextgarden.net/ConTeXt_Standalone`).

The resulting environment is quite usable. In fact this paper was partially written on this device.

To check how fast TEX is on the device, I used the following files:

1. `story.tex`: the famous story about Mr. Drofnats by A. U. Thor (see [9]).
2. The source of *The TEXbook*[9].[3]

[3] While TEXing of this book is prohibited, a special dispensation for benchmarking is traditionally recognized by the American Mathematical Society. I am grateful to Barbara Beeton for explaining this.

File	Pages	Engines				
		`tex`	`pdfetex`		`xetex`	`luatex`
			DVI	PDF		
`story.tex`	1	0.77	0.90	1.45	2.97	1.49
The TEXbook	494	5.84	6.94	11.63	12.05	18.50
LATEX 2_ε	492	N/A	23.84	26.28	29.10	32.66

Table 2: Benchmarks; times are in seconds

3. `source2e.tex`: the sources of LATEX 2_ε, as of 2011/06/27 [4].

All benchmarks were done with TEX Live 2011. Since only the `plain` format uses Knuthian TEX in this distribution, I benchmarked this engine only on the first two files. Also, LATEX requires several runs for the references to converge; only one run was measured. All benchmarks were done by calling

```
time TEX_COMMAND FILE
```

and taking the first time ('real time') from the output. Each test was repeated three times, and the average was taken.

The results are in Table 2. As seen from this table, a 500-page document is processed in about 15 seconds in plain and about 30 seconds in LATEX.

Another benchmark is compiling the present paper. One run of `pdflatex` takes 4.1 seconds.

The full compilation from scratch (issuing `make` after `make distclean`) involves a run of `pdflatex`, a run of `bibtex` and two runs of `pdflatex`.[4] This takes 13.3 seconds to complete. For comparison, on my desktop (4-core 2.4 MHz processor) it takes 1.9 seconds, and on my laptop (ASUS Eee PC 900HD, 800 MHz processor) — 11.9 seconds.

While details are beyond the scope of this paper, I would like to mention that such important and useful tools as R, Maxima, Octave, and Gnuplot also run on the device without any problems and with reasonable speed.

4 An alternative approach

This paper describes a creation of a full Linux environment under `chroot` on an Android device. Recently Mǎ Qǐ Yuán started to work on an alternative approach [11]: a compilation of TEX binaries using Android NDK. In this way one can create standalone applications that do not require a Linux installation to run. My tests showed that these applications are not faster than those under Linux; this is not surprising, since Linux applications are not run in a virtual machine, i.e., do not incur any overhead.

5 Conclusions

TEX and friends *can* run on an Android tablet. Moreover, they make it a useful work machine rather than a mere consumer toy.

Acknowledgements

I am grateful to my son Max Veytsman for rooting the device, helping with understanding the Android system, reading the manuscript and many useful comments; to Gaël Perron for help with Linux Installer; to Qǐ Yuán Mǎ for telling me about his approach; to Karl Berry and Barbara Beeton for encouraging this paper, suggesting benchmarking targets and editing the text.

References

[1] Anon. AsusTransformer Root + CWM recovery. `http://androidroot.mobi/technical/asus-eee-pad-transformer-tf101-root-cwm-recovery`, May 2011.

[2] AsusTeK Computer, Inc. Eee Pad Transformer TF101. `http://www.asus.com/Eee/Eee_Pad/Eee_Pad_Transformer_TF101`, 2011.

[3] Kaveh Bazargan. TEX as an eBook reader. *TUGboat*, 30(2):272–273, 2009. `http://www.tug.org/TUGboat/tb30-2/tb95bazargan.pdf`.

[4] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, and Rainer Schöpf. *The LATEX 2ε Sources*. LATEX3 Project, June 2011.

[5] Dangermouse. Gnome, KDE, IceWM or LXDE desktop on your Android! `http://www.androidfanatic.com/community-forums.html?func=view&catid=9&id=1615`, March 2009.

[6] GlavSoft LLC. TightVNC software. `http://www.tightvnc.com`, 2012.

[7] Patrick Hof. Installing a Debian chroot on the Asus Eee Pad Transformer. `http://www.offensivethinking.org/thoughts/2011/07/14/debian-chroot-eee-pad-transformer`, July 2011.

[8] Jonathan Kew and Stefan Löffler. TEXworks. Lowering the entry barrier to the TEX world. `http://www.tug.org/texworks`, 2012.

[9] Donald Ervin Knuth. *The TEXbook*. Computers & Typesetting A. Addison-Wesley Publishing Company, Reading, MA, 1994. Illustrations by Duane Bibby.

[10] TEX Live. Build procedure. `http://www.tug.org/texlive/build.html`, 2012.

[11] Qǐ Yuán Mǎ. TEX Live for Android. `http://code.google.com/p/texlive-for-android`, 2012.

[12] Jack Palevich. Android Terminal Emulator. `http://www.appbrain.com/app/android-terminal-emulator/jackpal.androidterm`, 2012.

[13] Gaël Perron. Linux installer. `http://android.galoula.com/en/LinuxInstall`, 2011.

[14] Phase Five Systems LLC. Jump Desktop. `http://www.jumpdesktop.com`, 2011.

[15] Valletta Ventures. The price of a messy codebase: No LATEX for the iPad. `http://vallettaventures.tumblr.com/post/13124883568/the-price-of-a-messy-codebase-no-latex-for-the-ipad`, November 2011.

[16] Denys Vlasenko. BusyBox. `http://busybox.net`, 2012.

[17] Tino Weinkauf and Sven Wiegand. TEXnicCenter — The center of your LATEX universe. `http://www.texniccenter.org`, 2012.

◇ Boris Veytsman
School of Systems Biology &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
USA
`borisv (at) lk dot net`

[4] As it happens, the second run is not necessary and is triggered by the message "Label(s) may have changed" produced by a too-cautious LATEX.