Play with Samsung Chromebook 3

BV

[2015-01-26 Mon 18:17]

1 From the Factory

1.1 Hardware

The Samsung chromebook 3 (model XE300C12) is a small, light and thin laptop with a full sized, if slightly unconventional, keyboard. It has an ARM processor (dual core ARMv7rev4 "v7l" Samsung EXYNOS5). It comes with 2GB RAM, 16GB eMMC flash. Interfaces USB2, USB3, HDMI and SD card and combined head phone / mic jack.

1.2 Software

It comes with Chrome OS which is Linux and a more or less common assortment of GNU and other user-land. But plopped on top of that is a UI based almost entirely on the Chrome browser. One can extend functionality by installing Chrome apps or extension. It is also possible to install Android apps: either those very few officially meant for this or, caveat user and in principle, any with chromeos-apk.

1.3 Usage

The expected usage paradigm it encourages is novel: the laptop is a web-based terminal to online services. Except for a "Downloads" directory, files are typically maintained on remote servers. Although, files can be saved locally this is not encouraged.

The authentication and authorization is also handled remotely. User's log in with Google credentials. With nominal usage patterns, one can be reasonably secure in having the laptop lost or stolen as nothing of importance actually resides on the "disk". 2-factor authentication only kicks in once on initial log-in so the device is then only as secure as your main Google password.

Multiple users are supported allowing a very useful way to securely loan out the laptop to others. By default any additional user can create an account or this may be restricted to a list allowed Google accounts created by the initial user.

1.4 Impressions

The speed of this laptop is impressive although it is heavily taxed by the weight of Chrome OS. Given the price (US$150 refurb), battery life (6'ish hours of real use), size, keyboard and even default software capabilities, this is by far the best laptop I've ever found in my two decades of laptop ownership. I won't be using it for detector simulations or large scale software development, but for web browsing, email, note taking while not paying attention to a meeting it is ideal.

But, what really drew me to Chrome OS was not Chrome OS but the ability to remove Chrome OS. Google has made it trivial to gain root access to the native OS and in fact encourages this as well as installing other OSes. This is done without compromising security for nominal use as gaining root ("developer mode" as it's called) and dropping it wipes out all trace of local files. This means an attacker gaining physical access can not use this mechanism to access your information nor surreptitiously plant malware. But, it does let people make full use of the system they bought.

1.5 A little Debian on the side

Because Chrome OS uses the Linux kernel it opens up the ability to run other Linux-based distributions in a `chroot`. This is best done using crouton. It allows multiple distributions to be run at the same time as Chrome OS, each on a different virtual console. The user can immediately switch between these different sessions as easily as switching between different virtual desktops. Crouton can be used without any disruption of Chrome OS and can be removed at any time. Only "developer mode" is needed.

Using this method, one quickly realizes that as fast as this little laptop is under Chrome OS, it's even faster in a Crouton image. It is a great way to have all the applications one is used to on a real OS while still making use of the benefits and simplicity of Chrome OS.

2 Going native

Using Crouton made me realize, I didn't want the benefits and simplicity of Chrome OS and as simple as it is to run Crouton images, I wanted to boot directly into a full Debian or Ubuntu OS. In part due to the ease Google has made using chromebooks and the creative talent of many individuals, there are various methods to install Debian or Ubuntu (or others) as the native OS.

2.1 Debian On

I first tried following the DebianOn page for this hardware. It targets installing to an SD card or USB stick and can culminate in a bit of media that you can plug in, boot and go to town. This was good to play with while fully preserving the internal flash with Chrome OS but I found SD card and (all I had at the time) USB 2 flash to be uncomfortably slow. And, because my intention was to use Debian/Ubuntu naively, having the OS run on external that could easily, if accidentally, be yanked at any time made me nervous.

2.2 ChrUbuntu

By far the best crafted method to install Ubuntu on a chromebook is ChrUbuntu. This is done by downloading and running a bash script as root under Chrome OS. It will leave the two Chrome OS boot/root partition pairs untouched and will use the 3rd pair to install. The space for this is taken from the remaining bulk of the internal flash which is meant for Chrome OS extensions and user data.

ChrUbuntu targets both ARM and AMD64 chromebooks and has gone through some evolution. As such, there seems to be some care needed to pick the best version to apply. I followed this post with the s9ryd version of the script (local copy). Running random scripts from the Internet as root is not generally advisable but this script is well written and easy to read. It is easy to assure oneself that nothing malicious is being attempted.

2.3 Comfort of familiarity

With the simplicity ChrUbuntu lends to getting the base OS up, the main effort is then in fixing up corner cases and installing my preferred desktop environment. The latter is somewhat responsible for increasing the number of the former. For me, this revolves around installing and configuring:

- MATE desktop environment
- Sawfish window manager
- Emacs

Rational (ie, my favorite) key bindings for the above are an extra challenge due to the somewhat unconventional keyboard.

2.4 Gory details

The rest of this document holds some curated notes on how I settled into a local maximum of satisfaction with this laptop.

3 Prepare

3.1 Developer mode

To start one must configure the "BIOS" of this laptop for "developer mode". This is done in one of the most deft manners. Hit the key combo: `esc-f3-power`, aka: `esc-[swirly]-[circlebar]`. This reboots the laptop and **wipes** the flash, replacing it (from some internal read-only copy) with a version of Chrome OS that has additional functionality and is referred to as *developer mode*. It takes ~10 minutes or so.

Once done, you are presented with the initial setup. Connect to WiFi and click through the EUL (Entirely Unreadable License) agreement. Then, stop and don't authenticate. Hit `ctrl-alt-f2`, aka `ctrl-alt-[->]` to get a Linux console. Log in as `chronos` with no password. You can "sudo -s" to gain root.

Wow, much control, so power, very root!

3.2 Installing with ChrUbuntu

The blog post is easy to follow. The steps involve:

1. Download and run the script to reformat boot/root pair 3 taking space from the general storage partition (I took largest recommended 9GB)
2. System will reboot to allow the "BIOS" to adjust the new partitions. It refers to this as "repairing" and will take several minutes.
3. Get back into the Linux console and re-download and re-run the script. It will notice the repartitioning is done and not repeat that

In this second running of the script you can control which Ubuntu distribution to install. With no arguments given it will install (at this time) Xubuntu 14.10. At one point I wanted to start as minimal as possible and so I tried `ubuntu-standard`. That ends up not installing the tools needed to get a working WiFi connection (`wireless-tools` and `wpa_supplicant` are missing). Adding the installation of these needed packages to the script could be done but it's easy enough to stick with the default Xubuntu as a starting point. Any unwanted packages can be pruned once a generally working system is achieved.

When all is said and done, one can log in as user and password "`user`" which can `sudo` to gain root. The first thing one should likely do upon login is to run:

```
$ sudo cgpt add -i 6 -P 5 -S 1 /dev/mmcblk0
```

This is needed in order to mark the Ubuntu boot/root partitions as having higher priority over the now vestigial Chrome OS ones. The exact command will be printed by the ChrUbuntu installation script just after installation completion and just before it reboots. If it's forgotten you can still enter it from a `chronos` shell on the Chrome OS side.

4 Have it your way

The rest of this document briefly goes over the end-game configuration to work around some issues that are likely common to anyone installing Ubuntu on a chromebook as well as to enact my own preferred environment. In doing this I've captured the commands in various "staged" shell scripts that can be run to automate repeating the configuration.