November 26, 2025

Chroming Up

Installing Linux On A Chromebook
To Write This Paper With

Contents

> Our informal corporate motto is "Don't be evil." We Googlers generally relate those words to the way we serve our users – as well we should. But being "a different kind of company" means more than the products we make and the business we're building; it means making sure that our core values inform our conduct in all aspects of our lives as Google employees.
>
> Google Code of Conduct [Feb 04 2004]

1 About

This paper will show the process and discoveries I have made while deploying Linux on an old Chromebook. This will include the process of unlocking the chromebook, installing linux (along with some customization options) and other miscellaneous configurations/discoveries.

You are ment to view this document by navigating to the topics in which you want to know. The table of contents will do wonders for this.

1.1 Why

For one reason, to allow you to do more things with this vender locked device. I would also prefer not to need a Google account to use my laptop.

Also for those of you who have not had the displeasure of being required to use a chromebook, may not know, these devices are locked down pretty tight for a laptop.

On a firmware level these devices prevent external tampering, and so, require some round about manor to do this devilish deed. To make life harder for us, these devices are, by default, hardware write protected, which dissalows any installation of an OS if you could acually access the storage portion of the device.

So ultimately, this is the practice of *hacking* in its original form. Making a computer do something it was not designed to do.

2 Getting Linux on a Chromebook

We will use `https://mrchromebox.tech`'s guide, firmware and utilities. First you need to find out if your device is supported. Before you get started you will need to get/check the following:

- Non ARM (x86_64 [Intel/AMD]) architecture
- The devices Hardware ID (HWID): found in the browser `chrome://system`
- A flash drive you can flash an OS to

- Determine if devices supports *UEFI Firmware (Full ROM)* and the *WP Method*[1]

I whent for *Replacing ChromeOS via Full ROM firmware*, so I will be using *UEFI Firmware (Full ROM)* method only. If you need to deviate you can just check mrchromebox.tech's docs for information on this process.

2.1 Developer Mode

Putting your device in *Developer Mode* "relaxes some of the restrictions in the *Verified Boot Mode*, and gives the user a bit more control over the system".[2]

To put your device in *Developer Mode* you will need to first enter *Recovery Mode*. It will vary by device. For *Chromebooks* you will need to "Press `ESC` + `Refresh` (arrow icon), + `Power` at the same time".[3]

After you have entered *Recovery Mode*, to get *Developer Mode* enabled, Press `[Ctrl+D` then `[ENTER]`, rebooting the device bringing up a screen telling your in "*Developer Mode*" or "OS Verification is OFF".

2.2 Disabling Write Protection

Next you will need to disable write protection for your computers drive, this differs pretty wildly, I have found. So hopefully you will already have found your device, and the write prtection method used on your model. The best resource I found is the chrultrabook project.[4]

2.3 Flashing Firmware

Once you have enabled *Developer Mode* and disabled *Write Protection*, then we can proceed to flashing *mrchromebox's* version of coreboot.

First you will need to get to a command line. You can do this by pressing "`[CTRL+ALT+F2]`" (or where `[F2]` would be on a keyboard). You can then login using the user `chronos`.

For the next step, *mrchromebox* has made a script to ease this process. To use it and flash the firmware, use the following commands.

```
cd
curl -LOf "https://mrchromebox.tech/firmware-util.sh"
sudo bash firmware-util.sh
```

[1] To check your device for support, and what Write Protection method, you can check out: https://docs.chrultrabook.com/docs/devices.html

[2] Getting *Developer Mode*: https://docs.mrchromebox.tech/docs/boot-modes/developer.html

[3] Getting to *Recovery Mode*: https://docs.mrchromebox.tech/docs/boot-modes/recovery.html

[4] To check your device for support, and what Write Protection method, you can check out: https://docs.chrultrabook.com/docs/devices.html

This will get you a little Terminal User Interface (TUI), that will show you multiple options. Since we are replacing the firmware, you will need to choose the "Install/Update UEFI (Full ROM) Firmware" option from the menu.

You should, probably, make a backup of your firmware, when prompted, but I did not. All thats left is to follow the prompts, and use the script to reboot your computer.

Congrates! You now have a liberated chromebook.

If you really want, you can customize your Coreboot. For example, you can change the rabbit.

3 Installing Linux

The installation of your linux distro sould be just as easy as if you were doing it on a normal computer. For this I chose Arch because it is not too hard, and yet doesnt hold your hand. It also now has an automated installer, which allows for much quicker, and smaller, installations of Linux on New computers.

But this guide is not for those who do not have the knowledge of such things.

3.1 Fixes and Problems

In mrchromebox's guide, he suggested that there would be more incompatibilities, but it, for the most part, works right out of the box for my model.

One notable, small, anoyence is that if I put the lid down, and it trys to suspend, it will not properly load back when it wakes.

3.1.1 Missing `initramfs-linux-lts.img`

After a couple of days of using and running this computer I encounter an error of which I have never seen before on this system: `initramfs-linux-lts.img not found`. The error itself is self evident, but how did it occur? I was not messing with any of the system configurations, especially not with the *initramfs*.

Spoiler alert, I fixed it. It turns out, probably (I am still unsure) that an update and a messed up configuration file seemed to prevent the *initramfs* from regenerating.

Booting into a live USB to do some diagnostics, I found that, indeed, there was no initramfs file. To fix this I chrooted[5] into the installation from the live USB, and regenerated my *initramfs*:

[5]The process I used to mount the guest OS to prepare for chrooting came from the Void Linux installation guide. https://docs.voidlinux.org/config/containers-and-vms/chroot.html#manual-method

```
# After you boot into your live USB you need to chroot into the
    installation.
# Here is an example of mounting and chrooting.

# These are Needed to have full functionality of the OS
mount -t proc  none   /mnt/proc
mount -t sysfs none   /mnt/sys
mount --rbind /dev    /mnt/dev
mount --rbind /run    /mnt/run

chroot /mnt

# Now to regenrate your initramfs, at least for Arch
# Yours may be different.
mkinitcpio -P
# The -P just processes all presets
```

When running that final command, it failed. It was due to invalid syntax in the `/etc/mkinitcpio.conf` file.[6] It seems that some how the file had been append invalid syntax. This fuels my theory that the upgrade process threw away the initramfs, because this configuration file failed.

The problem could have occured in relation to the one current problem that is occuring on this computer, since I had just forced shutdown the computer just before this problem started.

4 LaTeX

For writing this report I have choosen to use a familiar friend to me, LaTeX. This report has a little more convoluted process for developement than just running `pdflatex`. But for this Ill show you how to install it, and a basic editing setup.

[6] The default `/etc/mkinitcpio.conf` file can been seen in the Arch Linux github repository. https://github.com/archlinux/mkinitcpio/blob/master/mkinitcpio.conf

4.1 Necessary Dependencies

Since we are installing on Arch the following will be commands used with it Package Management Software (PMS) pacman. You will, however, need texlive-latex, a text editor, and a pdf viewer.

```
sudo pacman -Syu  # Base update/upgrade if a new system
# mupdf:   A common pdf viewer
# entr:    Run a command automatically when a file is updated
sudo pacman -S texlive-latex mupdf entr
```

You can create a basic LaTeXsetup using `entr` so you can automatically see your updates. With mupdf you can send the "HUP" signal to update its view. Using these basic configurations you can use a similar command to:

```
ls main.tex | entr -s 'pdflatex main.tex && pkill mupdf -HUP'
```

4.2 Further Possible Configuration

You can further set it up by redirecting the `entr -s` to a local script, and/or even put it in a script itself. You can use this method, along with the `\input{}` LaTeXcommand, to create a seperate file that has a dynamic data in it.

You could also use pandoc, to convert a markdown file to TeXand then include it. This method can become monstrous and beautiful.

You can learn more about how to use LaTeXhere:

- https://latex-tutorial.com/tutorials/
- https://www.overleaf.com/learn

This report contains aproxamently 1380 words.