

GNU Boot manual (version 496ad2f)

GNU Boot Contributors (gnuboot@gnu.org)

Copyright © 2023-2026 Denis 'GNUtoo' Carikli.

Copyright © 2024 Adrien 'neox' Bourmault.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Table of Contents

1	Overview	1
1.1	What is GNU Boot	1
1.1.1	boot software	1
1.1.2	distribution	2
1.2	Why free boot software is important	3
1.3	Why use GNU Boot	3
1.3.1	How much free software is GNU Boot?	3
1.3.2	Limitations	4
1.4	Other free boot software distributions	5
1.4.1	GNU boot and GNU Guix	5
1.4.2	GNU boot and Canoeboot	5
1.4.2.1	GNU boot and Canoeboot: features	5
1.4.2.2	GNU boot and Canoeboot: fixes	6
1.4.2.3	GNU boot and Canoeboot: dependence on upstream	6
1.4.3	How much free software are other distributions	7
2	Supported hardware and configurations	9
2.1	Supported computers	9
2.1.1	Refurbished and/or modified computers	10
2.1.2	Experimental support for some computers	10
2.1.3	Asus KGPE-D16 (experimental)	12
2.1.3.1	Asus KGPE-D16: How to build a computer with this mainboard	12
2.1.4	Lenovo ThinkPad X60	12
2.1.4.1	Lenovo ThinkPad X60 builtin card reader	13
2.1.5	Lenovo ThinkPad X60s	13
2.1.5.1	Lenovo ThinkPad X60s builtin card reader	13
2.1.6	Lenovo ThinkPad X60T	13
2.1.6.1	Lenovo ThinkPad X60T builtin card reader	13
2.1.7	Lenovo ThinkPad X200	13
2.1.7.1	Lenovo ThinkPad X200 builtin boot flash	14
2.1.7.2	Lenovo ThinkPad X200 builtin CPU	14
2.1.7.3	Lenovo ThinkPad X200 builtin card reader	14
2.1.8	Lenovo ThinkPad X200s	14
2.1.8.1	Lenovo ThinkPad X200s builtin card reader	14
2.1.9	Lenovo ThinkPad X200T	14
2.1.9.1	Lenovo ThinkPad X200T builtin card reader	15
2.1.10	Libiquity Taurinus X200	15
2.1.10.1	Taurinus X200 builtin card reader	15
2.1.11	Technoethical X200	15
2.1.11.1	Technoethical X200 builtin card reader	15
2.1.12	Technoethical X200s	16
2.1.12.1	Technoethical X200s builtin card reader	16

2.1.13	Technoethical X200 Tablet (X200T)	16
2.1.13.1	Technoethical X200T builtin card reader	16
2.1.14	Vikings X200	16
2.1.14.1	Vikings X200 builtin card reader	17
2.2	Supported computer parts and peripherals	17
2.2.1	PCI and PCIe cards with software	17
2.2.2	Supported GPUs and graphics	18
2.2.3	Supported external card readers	19
2.2.4	Supported serial ports	19
2.2.5	Unsupported hardware supported by projects reused by GNU Boot	19
2.3	Supported operating systems	20
2.3.1	GNU/Linux	20
2.3.1.1	Non-standard GNU/Linux configurations	20
2.3.1.2	Nonfree distributions	20
2.3.2	Other operating systems	20
2.4	GNU Boot images	21
2.4.1	GNU Boot images naming	21
2.4.2	GNU Boot images types	22
2.5	SeaBIOS images requirements	23
2.5.1	SeaBIOS: Supported filesystems and partitions	23
2.5.2	SeaBIOS: Supported bootloaders	23
2.5.2.1	SeaBIOS: GRUB (on MBR or BIOS boot partition) ...	23
2.6	GRUB images requirements	26
2.6.1	GRUB: Supported configuration files	26
2.6.1.1	grub.cfg	26
2.6.2	GRUB: Supported partitions	26
2.6.2.1	GRUB: GPT (GUID Partition Table) and MBR (Master boot record)	26
2.6.2.2	GRUB: LVM2	26
2.6.2.3	GRUB: GNU/Linux RAID	26
2.6.2.4	GRUB: RAID formats of RAID card vendors	26
2.6.2.5	other partition types	26
3	Flash programmers	27
3.1	Practical freedoms with flash programmers	27
3.2	Setup and components needed to use a flash programmer	27
3.3	Flash chip package	28
3.4	Known issues with flash programmers	28
3.4.1	Wire length	28
3.4.2	Programmer SPI speed	28
3.4.3	Lenovo ThinkPad X200 with the MX25L6405D (8MiB)	28
3.5	Supported flash programmers	29
3.5.1	OpenMoko Neo1973 Debug board (V2+)	29
3.5.2	Vikings CH341a USB Programmer	29
3.5.3	Zerocat Chipflasher v2 and its variations	29

3.5.3.1	Zerocat Chipflasher v2.....	30
3.5.3.2	Zerocat Chipflasher v2 without builtin firmware.....	30
4	Installing or upgrading GNU Boot images ...	32
4.1	Installation and upgrade instructions	32
5	Using GNU Boot	33
5.1	Using GNU Boot with QEMU	33
5.1.1	Quick test with QEMU	33
5.1.2	Bootng a distribution with QEMU	34
5.1.3	Bootng an x86 64bit distribution with QEMU	37
5.1.4	Bootng more distributions with QEMU	38
6	Security	39
6.1	Very basic introduction on security	39
6.1.1	Threat modelling or what security really is about	39
6.1.2	Impact of malware	40
6.2	Security considerations when using GNU Boot	41
6.2.1	Secure boot and restricted boot	42
6.3	Security considerations when updating GNU Boot	42
6.3.1	Checking the signatures of GNU Boot images	43
6.4	Security considerations when installing GNU Boot	44
6.4.1	Installing from a computer with a nonfree BIOS	44
6.4.2	Installing from a computer with another boot software distribution	44
6.4.3	Installing with a 100% hardware flash chip programmer	45
6.4.4	Installing with a flash chip programmer that has a firmware.	45
6.5	Trusting hardware	46
7	Maintenance and repairs	47
7.1	Replacing the boot flash on a desktop, server or workstation	47
8	Building GNU Boot from source	48
8.1	Supported distributions for building GNU Boot binaries	48
8.2	Installing Guix	49
8.2.1	Enabling Guix substitutes	49
8.2.2	Enabling Guix all substitutes servers	50
8.2.3	GNU Boot configuration for Guix	51
8.2.4	Managing space with Guix	51
8.3	Building from tarballs and/or offline	51
8.4	Downloading the GNU Boot source code with git	51
8.5	Authenticating the GNU Boot source code	51
8.6	Initializing the GNU Build system	52
8.7	Configuring the GNU Boot build	53

8.8 Installing dependencies for building GNU Boot	54
8.9 Building a GNU Boot release	54
9 Helping GNU Boot	56
Appendix A GNU Free Documentation	
License	57
Concept index	65

1 Overview

This chapter will explain what is GNU Boot, and how it compares with somewhat similar projects.

1.1 What is GNU Boot

GNU Boot is a boot software distribution. What this means will be explained below.

1.1.1 boot software

When you boot a computer, you can sometimes see a screen that looks like that:

```
+-----+
|                                             |
|                                             |
|                                             |
|                                             |
|               [ Some company Logo ]               |
|                                             |
|                                             |
|                                             |
|                                             |
| Press F2 for BIOS setup, Press F12 for the startup menu. |
+-----+
```

This screen is produced by software that is often called BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface). We refer to this type of software as boot software.

The goal of this software is to initialize the hardware of the computer and load an operating system (like GNU/Linux, Windows, or macOS). GNU Boot can replace this software on some computers.

Like all software, this boot software has to be stored somewhere.

Unlike the operating system which is typically stored on data storage devices people are familiar with (like SSD (Solid State Drive) or a hard disks, USB keys, etc), the boot software is stored on a very different kind of data storage device.

It is stored inside a chip that is inside the computer mainboard. People and technical documentation often call this chip the *boot flash* because it stores the boot software, and also because it is a *flash chip* which is a chip that stores data and that is built with a semiconductor technology called flash (see the Wikipedia article on flash memory ([https://en.wikipedia.org/wiki/Flash memory](https://en.wikipedia.org/wiki/Flash_memory)) for more details on the technology).

Here is a picture of this boot flash on the mainboard of a ThinkPad X200:



Boot software (like BIOS (Basic Input/Output System), UEFI (Unified Extensible Firmware Interface) or GNU Boot) exist for a variety of technical reasons:

- The operating systems require certain hardware components like the RAM (Random Access Memory) to already work when they are started.
- The operating system is stored on a storage device(s) (like SSD (Solid State Drive), hard disks, etc) and part of it needs to be loaded inside the RAM (Random Access Memory) to work. Something has to do the loading, and this is done in software for flexibility and/or efficiency reasons.
- Finally, certain hardware components cannot be auto-detected and something needs to tell the operating system what drivers to load, which which settings.

GNU Boot provides such software. It enables to replace nonfree boot software (typically nonfree BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface)) on some computers.

1.1.2 distribution

GNU Boot a distribution of boot software: like GNU/Linux distributions (like Trisquel), it reuses various software to produce something that can be installed.

However the scope of GNU Boot is more limited than GNU/Linux distributions as it only reuses boot software like Coreboot or GRUB to produce something that replaces nonfree BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface).

1.2 Why free boot software is important

Freedom is important in general, and running nonfree software has negative consequences regardless of the type of software (game, boot software, operating system, driver, etc).

Here are some examples of common issues for nonfree boot software:

- Since the boot software loads the operating system, it can potentially modify it in a malicious way. In most cases part of the boot software also continues to run once the operating system is started. Because of that and, and because of the way the hardware and boot software run, the boot software can also do such modification at any time. If the boot software is nonfree, it is way harder to find and remove malicious code (it's even impossible to remove in some cases), and there is no way to make sure that there is none left. For instance many nonfree boot software were shipped with the CompuTrace malware (which was advertised as an anti-theft security feature).
- Vendors of various hardware components have to collaborate together to provide updates for nonfree Boot software, so in practice they decide when updates are done. So if a computer is not sold anymore, it is unlikely to get update for its Boot software unless the Boot software uses some free software that can be updated. Also note that applying nonfree updates comes with huge risk as we don't know what's inside the updates. Hardware vendors who provide the updates also have an incentive to make things worse for the users, so they would be pushed to buy new devices.
- Some nonfree Boot software restrict what you can do with your computer. For instance they refuse to boot if you changed or removed some hardware components.

1.3 Why use GNU Boot

As explained before GNU boot is just a distribution. So it is also possible to take the same software that GNU Boot reuses, and to build, assemble and install it yourself.

However doing that is risky because if something goes wrong, your computer won't boot anymore.

So the goals of GNU Boot are to:

- Collaborate together to test if GNU Boot releases works fine.
- Provide documentation to enable easy installation and usage.
- Limit the amount of work done by GNU Boot and contribute directly to the software we reuse whenever possible.

GNU Boot also has a long term focus, so it tries not to break users use cases, and tries as much as possible to fix issues in the projects it reuses instead of doing workarounds that impact users.

1.3.1 How much free software is GNU Boot?

Being a GNU package, GNU Boot itself has to be 100% free software.

If you find nonfree software in GNU Boot and/or any source code or binaries released by GNU Boot, please contact its maintainers by opening a bug report on its bug tracker at <https://savannah.gnu.org/bugs/?group=gnuboot>.

But that doesn't mean that GNU Boot magically makes everything not provided by GNU Boot free software. For instance if you install a nonfree operating system, GNU Boot will try to run it.

In some cases GNU Boot even runs nonfree software not provided by GNU Boot like nonfree GPUs drivers provided by the removable GPU card. See Section 2.2 [Supported computer parts and peripherals], page 17, for more details about this issue and how to avoid running such nonfree software.

To address problems like that the Free Software Foundation (<https://www.fsf.org/>) has created the Respect Your Freedom hardware certification (<https://ryf.fsf.org/>) to list hardware that works with only free software (with some very small exceptions for some components, see its criteria (<https://ryf.fsf.org/about/criteria>) for more details).

In addition there is also The Blob Fallacy article (<https://www.fsfla.org/ikiwiki/blogs/lxo/draft/blob-fallacy>) or a video of a presentation about the same issue at LibrePlanet 2024 (<https://media.libreplanet.org/u/libreplanet/m/software-enshittification-or-freedom-it-s-not-a-hard-choice>) by Alexandre Oliva that explains the related freedom issues with nonfree software provided by the hardware and how they compare with other kind of freedom issues (nonfree driver, nonfree firmware loaded automatically by Linux, etc).

If we dig deeper, hardware supported by GNU Boot is also affected by additional issues. They are less problematic than the rest as the Respect Your Freedom hardware certification makes exception for these until we get free software replacements.

All the (mass) storage devices (Hard disks, SSD (Solid State Drive), USB keys, SD cards, etc) have a builtin nonfree firmware. It is possible to avoid trusting these nonfree firmware by encrypting everything that is on these devices. How to do that will need to be added later on in this manual.

In the case of laptops, there is however something more problematic for which we have no solution yet. The computers have a builtin nonfree firmware which handle the keyboard (which is extremely problematic), and various tasks related to power management (setting up the right voltages at boot, managing the battery, cutting power to the WiFi, the power management beeps, etc).

In that case the way to go would be to find a way to make sure that people do work on addressing the issue by writing a free firmware to replace the nonfree firmware, and by finding a way to upgrade the nonfree firmware to a free one.

The Free Hardware and Free Hardware Designs (<https://www.gnu.org/philosophy/free-hardware-designs.html>) also has good background on these topics as in some case hardware and firmware is indistinguishable, and so this also has consequences on freedom and/or on how to recognize the most problematic issues, which in turn is a useful tool for organizing the free software movement at large.

1.3.2 Limitations

GNU Boot is fairly recent and doesn't have an official release yet.

For the release we plan to have at least some install and upgrade instructions for some computers and an easy way for users to use GNU Boot.

Also the latest GNU Boot release candidate was not tested yet with all the computers it's supposed to support (we badly need help for that).

1.4 Other free boot software distributions

The following GNU/Linux distributions should also provide 100% free boot software but they usually only provide them for computers using the ARM architecture (which GNU Boot doesn't support yet):

- Parabola
- PureOS
- Trisquel

1.4.1 GNU boot and GNU Guix

The GNU Guix package manager (which GNU Boot also reuses) also provide 100% free boot software for some ARM computers. However the Guix packages are updated all the time and the Guix project doesn't provide any way for users to report that specific ARM computers work fine with the boot software they provide.

1.4.2 GNU boot and Canoeboot

There is also Canoeboot which is a 100% free software boot distribution similar to GNU Boot. Its goal is to remove nonfree software from Libreboot.

The main difference between both is that GNU Boot has a longer term focus.

This has some wide reaching consequences that affect many areas. Because of that, the next subsection will give a few practical examples to show this in practice rather than trying to be exhaustive.

Also note that the GNU Boot maintainers don't know Canoeboot well enough as this would require them to spend a lot of time studying its documentation, source code and testing it and/or using it, to get first hand knowledge about it, to really evaluate the impact of Canoeboot choices and compare them to GNU Boot.

So instead, the GNU Boot maintainers relied on a mix of discussion with Leah Rowe, the Canoeboot maintainer, and their own research, to get information about Canoeboot.

1.4.2.1 GNU boot and Canoeboot: features

Compared to GNU Boot, Canoeboot focuses more on having the latest software, adding additional features not available in the projects it reuses (like modern full disk encryption with GRUB), and it also supports computers not yet supported by GNU Boot.

In both cases this is sometimes done by modifying the project it reuses without contributing back the changes. While this saves a lot of time in the short and medium term, and that things seems to work as, according to its maintainer, Canoeboot and/or its community do test all these changes, it also makes it harder to keep these extra features and/or computers in the long run.

As the project being reused and everything around evolve, the modifications need to be maintained and sometimes re-made to work with newer versions of the project being reused.

The difficulty to keep these changes increase with the amount of such changes, their size and complexity, or the speed of the evolution of the projects being reused.

Weather or not this will turn out to be an issue is hard to tell in advance because there are a lot of unknowns.

Note that GNU Boot also has similar issues but instead of adding more modifications it tries to reduce their amount.

1.4.2.2 GNU boot and Canoeboot: fixes

Another significant difference between both projects is that Canoeboot seems to have fixed a lot of issue since Libreboot 20220710 (the latest fully Libreboot version), while GNU Boot took a completely different approach: its goal is to re-architecture many things, step by step, while taking great care of not breaking what worked in Libreboot 20220710. For instance GNU Boot is switching to GNU Guix to build its software and it is also making a proper manual.

In the long run this enables GNU boot to solve more complex problem that would be really hard or impossible to solve with the current design of Canoeboot (like the ability to blindly trust the images built by someone else, or the ability to build and modify your own images faster, more easily and with way less risk) or have more standard documentation in the form of a manual.

But this also means that shorter term fixes are often less good in GNU Boot because its contributors focus more on longer term fixes like the migration to GNU Guix.

The real impact of all that is hard to evaluate in the usage of Canoeboot or GNU Boot in the short term as GNU Boot also changes less often than Canoeboot, and GNU Boot reports tests on its status page, while there is no such things for Canoeboot (however if things don't work people can also report bugs).

As for the impact on the documentation, it is also somewhat similar: In GNU Boot, the information is being moved from the website to the manual, to be more consistent and easier to understand, but this is also done at the expense of the website which has fewer contributions.

In Canoeboot the website is being improved instead. And here too, GNU Boot also has tracks the documentation through a scary banner on the website pages that weren't reviewed.

In both cases this can enable GNU Boot to find issues and revisit many things to improve a lot the quality in the longer run, at the expense of quicker fixes and improvements. Canoeboot also does and/or did that in one way or another but there is no clear separation of what was reviewed or not.

1.4.2.3 GNU boot and Canoeboot: dependence on upstream

Another difference is that nowadays the code and website of Canoeboot are closer to the current Libreboot than GNU Boot is.

In practice it means that a lot of the work from Libreboot is reusable in Canoeboot with way less effort, but the downside is also that it is harder to tailor Canoeboot to the needs and habits of users and contributors that don't want to run nonfree software.

An example is that at the time of writing, the most well tested distributions to build Canoeboot are not endorsed by the Free Software Foundation (<https://www.fsf.org/>).

It could be fixed but this would require either the maintainer or dedicated contributors to test everything with these distributions, and in this case, Canoeboot would not rely on the testing already done in Libreboot with non-endorsed distributions.

Another way would be to require Libreboot to test everything with endorsed distributions as well, or to use endorsed distributions that are as close as possible to the most well tested distributions to build Canoeboot.

In GNU Boot, there is the opposite situation: the project only requires to test with endorsed distribution(s), but it is also possible to install the most supported ones on top of another distribution with Guix’s debootstrap (but this is not documented). The longer term fix in GNU Boot is to build everything in Guix, which can be installed more easily on top of many distributions.

This difference of distributions support is reflected in the documentation of both projects as well, which assumes that users use or want to use certain distributions.

Note that the pages of the GNU Boot website that weren’t reviewed (and that have a scary warning because of that) might still assume that users use nonfree distributions.

1.4.3 How much free software are other distributions

Apart from Parabola for ARM computers that only officially support computers that can boot with free software, all other distributions, which also includes Parabola for x86 computers also support computers that boot with nonfree software: these computers come with nonfree software pre-installed and these distributions reuse that software to boot.

So the fact that a computer is supported by these distributions does not indicate that the computer can boot with fully free software.

Here are some examples for the GNU/Linux distributions. They all provide packages that only work with nonfree software that is already on the computer.

Distribution	Package	Support
Guix	u-boot-rpi-arm64	Works only with nonfree Raspberry PI bootloader
Parabola (x86)	tp_smapi	Works only with nonfree ThinkPad BIOS
PureOS	u-boot-rpi	Works only with nonfree Raspberry PI bootloader
Trisquel 11 (aramo)	u-boot-rpi	Works only with nonfree Raspberry PI bootloader

As for Canoeboot, since June 2025, it also supports computers that boot with nonfree software. This nonfree software runs in a separate computer within Intel computers that is called Management Engine.

On supported computers, this nonfree software, has access to the network and can take full control of the computer at any time (even once booted or when the computer seems to be powered off). This is even advertised by the Intel (the CPU manufacturer) as a feature to more easily control computers remotely.

See the The Intel Management Engine: an attack on computer users' freedom (<https://www.fsf.org/blogs/sysadmin/the-management-engine-an-attack-on-computer-users-freedom>) article has more information on this issue.

To limit the damage, Canoeboot kindly asks this nonfree software to not run the most problematic part of code (this includes the code that is responsible for the remote control) but also doesn't remove any of this code. Since this operating system is nonfree it is hard to really understand the impact of doing that.

You may also be able to completely remove the most problematic part of this code yourself with `me_cleaner` (https://github.com/corna/me_cleaner), but it is not possible to remove all of this nonfree code (otherwise Canoeboot would have done that). Here too it is hard to really understand the impact of running the remaining nonfree code to boot the computer.

Really understanding the impact is extremely complex or impossible, basically because the code is nonfree.

If need more details, there is some documentation on what these settings are supposed to do, like the Disabling Intel ME 11 via undocumented mode (<https://web.archive.org/web/20170828150536/http://blog.ptsecurity.com/2017/08/disabling-intel-me.html>) article by Positive Technologies, or the source code, documentation, bug reports and contributions of the `me_cleaner` project. All that shows that some nonfree code still run at boot, and all the affected computers don't necessarily run the exact same nonfree code, and research often analyze specific versions of this nonfree code, and not necessarily the code that you might be running if you use such computers with Canoeboot. In addition research tend to analyze specific part of the code only and not everything.

Because of that, really knowing the danger of running the nonfree software code that is required to boot the computer is almost impossible, because even if you do a lot of research yourself, this would still need to be somehow reviewed by peers to get a bit more assurance it doesn't contain huge mistakes.

So given the capabilities that kind of hardware and software has, it is a good idea to be extra cautious. So it would be best to avoid relying on computers with such issues if you can.

Note that this issue is not limited to Intel, as recent CPU made by AMD also have somewhat similar issues. Most smartphones also have a similar issues, and even some ARM computers like the Raspberry Pi also have a similar issue. See the What's wrong with the Raspberry Pi (<https://web.archive.org/web/20220531105341/https://ownyourbits.com/2019/02/02/whats-wrong-with-the-raspberry-pi/>) for more details on the Raspberry Pi.

2 Supported hardware and configurations

2.1 Supported computers

For now, GNU Boot only provides images that can be installed on the following computers:

- Acer G43T-AM3
- Apple MacBook 1.1
- Apple MacBook 2.1
- Apple iMac 5,2
- Asus KCMA-D8 (experimental)
- Asus KFSN4-DRE (experimental)
- Asus KGPE-D16 (experimental)
- Gigabyte D945GCLF2D
- Gigabyte GA-G41M-ES2L
- Intel D410PT
- Intel D510MO
- Intel D945GCLF
- Lenovo ThinkPad R400
- Lenovo ThinkPad R500
- Lenovo ThinkPad T400
- Lenovo ThinkPad T400S
- Lenovo ThinkPad T500
- Lenovo ThinkPad T60 with intel GPU
- Lenovo ThinkPad W500
- Lenovo ThinkPad X200
- Lenovo ThinkPad X200S
- Lenovo ThinkPad X200T
- Lenovo ThinkPad X301
- Lenovo ThinkPad X60
- Lenovo ThinkPad X60T
- Lenovo ThinkPad X60s
- Libiquity Taurinus X200
- Qemu PC (i440FX)
- Technoethical D16 (experimental)
- Technoethical T400
- Technoethical T400s
- Technoethical T500
- Technoethical X200
- Technoethical X200s

- Technoethical X200 Tablet (X200T)
- Vikings ASUS KCMA D8 mainboard and workstation (experimental)
- Vikings ASUS KGPE D16 mainboard (experimental)
- Vikings X200

However as GNU Boot is still relatively new, we lack installation and upgrade instructions for most of these computers.

Also not all are well tested, so it's a good idea to look on the GNU Boot website, on the status page (<https://www.gnu.org/software/gnuboot/status.html>) for up to date result of tests by GNU Boot users and contributors.

2.1.1 Refurbished and/or modified computers

Note that, while they are very similar, a Lenovo ThinkPad X200 is not the same thing than a Technoethical X200. This is because vendors like Technoethical, Vikings, and others sometime modify the hardware and/or software of the computers they sell. Note that some of these vendors sold different computer models and not just the X200.

At least one of these vendor (Minifree Ltd) is known to have increased the boot flash size, and multiple vendors are also known to have modified the boot software images they shipped to make sure that the builtin Ethernet controller hardware address (also known as MAC address (Medium Access Control address)) was the same than on the sticker below the computer.

Since these modifications are useful but that they can have impacts in multiple areas (installation or repair instructions, flash chip programmer compatibility, etc), we chose to take into account these differences to best support these potentially modified computers, and the way we did it is by considering the Lenovo ThinkPad X200 and the Technoethical X200 to be two distinct computers.

Because of that we also need help from the vendors and/or users of these modified computers to tell us or help us find what was modified on these computers, to better support them.

2.1.2 Experimental support for some computers

The support for the following computers is experimental:

- Asus KCMA-D8
- Asus KFSN4-DRE
- Asus KGPE-D16
- Technoethical D16
- Vikings ASUS KCMA D8 mainboard and workstation
- Vikings ASUS KGPE D16 mainboard

If you don't have any of these computers, you can simply skip this subsection.

All of these computers use similar chips. Their future is uncertain because of several issues:

- They are no longer supported by Coreboot. This makes it difficult continue supporting it over time. In practice they uses an older Coreboot version and the code doesn't build

on distributions like Parabola because of that. For now the code can still be built on Trisquel 11 (aramo).

- The code that initialize the RAM works accidentally and it is not reliable. In practice this means that the compatibility RAM modules is very difficult to know in advance. For instance going up to the maximum amount of RAM (like 256 GiB of RAM on a KGPE-D16) only works with very specific RAM modules.

Which modules work and can still be bought at regular prices are also kept secret to prevent companies from buying them all and increasing the prices.

In addition sometimes rebooting don't work, and powering the mainboard does not always work (sometimes we need to try several times for that to work). Weather that works or not can also be dependent on the room temperature. In addition once running the computer for very long period of time can result in a crash at some point, which requires rebooting it.

Despite all that, some organizations like the FSF, GNU or Libre en Communs manage to run infrastructure with computers that have these issues. But this requires specific know how or workarounds.

For instance if you want to use these computers as remote servers, you will need to know how to build a system that can reboot the computer remotely even if it is crashed, and also make sure that the system you build doesn't depend on nonfree software.

In contrast, such knowledge is not required for running GNU Boot on well supported computers like a ThinkPad X200, even if they are used as servers.

Despite all these issues, these computer that are affected by such issues are still (badly) supported by GNU Boot because of historic reasons: in November 2022, Libreboot began to include non-libre code, and because of that GNU Boot was created to continue providing a fully free boot distribution with the same spirit of former Libreboot releases.

Since Libreboot already supported this computer and that people and organization do use them, including key contributors to GNU Boot, it made sense and still makes sense to support these mainboard.

For instance the GNU Boot build server runs on a VM (virtual machine) provided by Libre en Communs which runs on a KGPE-D16 with GNU Boot, and the KGPE-D16 is affected by these issues.

And neox is also involved in running all that on the Libre En Communs side so not supporting that mainboard in GNU Boot would also means that neox would have less time for GNU Boot due to the duplication of work.

All the above makes it really important to fix these issue once for good, and several things can be done to really do that:

- neox reverse engineered the RAM controller hardware of the KGPE-D16 and wrote an academic paper on it that will be published under a free license. People with related skills can help by requesting a copy to neox and reviewing it.
- another way to help is to find people that have the skills and the time to write a proper RAM initialization code and bring back this mainboard in Coreboot. GNU Boot may be able to get funding for that.

In addition fixing these issue will also enable to have RAM initialization code that we have the full control of: most computers out there put unknown values in the RAM controller and have algorithms to then tune the RAM settings until something acceptable is reached.

But the goal of neox's paper is to really document the hardware, to enable to write extremely reliable initialization code. This can also open the door for other optimizations to improve performance, reliability (make it work with a bigger temperature range), security, etc.

These optimizations are not possible for all the other computers supported by GNU Boot, as it would require to do the same kind of work neox did to also reverse engineer the RAM controller of these other computers.

2.1.3 Asus KGPE-D16 (experimental)

Here are some details on this computer:

Vendor: Asus
 Product: KGPE-D16
 Max RAM: 256 GiB (with specific modules)
 CPU: 2 CPUs, Removable: G34 socket
 Boot flash: Removable: DIP-8 socket

This computer can have 32 cores and 256 GiB of RAM, so it is the most powerful x86 computer that can boot with free software. More x86 recent computers require nonfree software to boot.

It also exist more powerful computers that can boot with free software, but the support for fully free distribution is either experimental or non-existent depending on the computer. In addition these more powerful computers are much more expensive, limiting their reach to organizations or individuals with enough money to buy them (some people in the free software community have such computers, but not a lot of people).

The support for this computer is experimental. See Section 2.1.2 [Experimental support for some computers], page 10, for what it means.

2.1.3.1 Asus KGPE-D16: How to build a computer with this mainboard.

At the moment, USB keyboards are extremely slow in the GRUB provided by GNU Boot in its GRUB images. PS/2 Keyboard seems to work fine, so it is strongly recommended to get a PS/2 keyboard or try to see if some USB to PS2 adapters can work. A PS/2 keyboard was tested and worked fine when connected on the purple PS/2 connector.

2.1.4 Lenovo ThinkPad X60

Here are some details on this computer:

Vendor: Lenovo
 Product: ThinkPad X60
 Max RAM: < 4GiB
 CPU: Non-removable (soldered)
 Boot flash: Non-removable (soldered)

2.1.4.1 Lenovo ThinkPad X60 builtin card reader

The internal SD card reader is known not to work with GNU Boot but it works with Linux. This means that you cannot boot a distribution from an SD card or a microSD card with the internal SD card reader. However once booted, Linux has full access to the SD card and it can read very low level technical information about the cards (like their serial number, vendor, etc) or support SD peripherals (like TV tunners, etc).

2.1.5 Lenovo ThinkPad X60s

Here are some details on this computer:

Vendor: Lenovo
Product: ThinkPad X60s
Max RAM: < 4GiB
CPU: Non-removable (soldered)
Boot flash: Non-removable (soldered)

2.1.5.1 Lenovo ThinkPad X60s builtin card reader

The internal SD card reader is known not to work with GNU Boot but it works with Linux. This means that you cannot boot a distribution from an SD card or a microSD card with the internal SD card reader. However once booted, Linux has full access to the SD card and it can read very low level technical information about the cards (like their serial number, vendor, etc) or support SD peripherals (like TV tunners, etc).

2.1.6 Lenovo ThinkPad X60T

Here are some details on this computer:

Vendor: Lenovo
Product: ThinkPad X60T
Max RAM: < 4GiB
CPU: Non-removable (soldered)
Boot flash: Non-removable (soldered)

2.1.6.1 Lenovo ThinkPad X60T builtin card reader

The internal SD card reader is known not to work with GNU Boot but it works with Linux. This means that you cannot boot a distribution from an SD card or a microSD card with the internal SD card reader. However once booted, Linux has full access to the SD card and it can read very low level technical information about the cards (like their serial number, vendor, etc) or support SD peripherals (like TV tunners, etc).

2.1.7 Lenovo ThinkPad X200

Here are some details on this computer:

Vendor: Lenovo
Product: ThinkPad X200
Max RAM: 8GiB
CPU: Non-removable
 (soldered)
Boot flash: Non-removable
 (soldered)

2.1.7.1 Lenovo ThinkPad X200 builtin boot flash

The ThinkPad X200 originally had one of these boot flash:

- Macronix MX25L6405D (8MiB, SOIC-16)
- Winbond W25X64 (8MiB)
- Macronix MX25L3205D (4MiB)
- Atmel AT25DF321 (4MiB)

2.1.7.2 Lenovo ThinkPad X200 builtin CPU

The ThinkPad X200 has one of these CPUs:

- Intel Core 2 Duo P8400 (64bit, 2 cores, 2.26GHz)
- Intel Core 2 Duo P8600 (64bit, 2 cores, 2.40GHz)
- Intel Core 2 Duo P8700 (64bit, 2 cores, 2.53GHz)
- Intel Core 2 Duo P8800 (64bit, 2 cores, 2.66GHz)

2.1.7.3 Lenovo ThinkPad X200 builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.8 Lenovo ThinkPad X200s

Here are some details on this computer:

Vendor:	Lenovo
Product:	ThinkPad X200s
Max RAM:	8GiB
CPU:	Non-removable (soldered)
Boot flash:	Non-removable (soldered)

2.1.8.1 Lenovo ThinkPad X200s builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.9 Lenovo ThinkPad X200T

Here are some details on this computer:

Vendor:	Lenovo
Product:	ThinkPad X200%
Max RAM:	8GiB

CPU: Non-removable
(soldered)
Boot flash: Non-removable
(soldered)

2.1.9.1 Lenovo ThinkPad X200T builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.10 Libiquity Taurinus X200

Here are some details on this computer:

Vendor: Libiquity
Product: Taurinus X200
Max RAM: 8GiB
CPU: Non-removable
(soldered)
Boot flash: Non-removable
(soldered)

2.1.10.1 Taurinus X200 builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.11 Technoethical X200

Here are some details on this computer:

Vendor: Technoethical
Product: X200
Max RAM: 8GiB
CPU: Non-removable
(soldered)
Boot flash: Non-removable
(soldered)

2.1.11.1 Technoethical X200 builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.12 Technoethical X200s

Here are some details on this computer:

Vendor:	Technoethical
Product:	X200s
Max RAM:	8GiB
CPU:	Non-removable (soldered)
Boot flash:	Non-removable (soldered)

2.1.12.1 Technoethical X200s builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.13 Technoethical X200 Tablet (X200T)

Here are some details on this computer:

Vendor:	Technoethical
Product:	X200T
Max RAM:	8GiB
CPU:	Non-removable (soldered)
Boot flash:	Non-removable (soldered)

2.1.13.1 Technoethical X200T builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.1.14 Vikings X200

Here are some details on this computer:

Vendor:	Vikings
Product:	X200
Max RAM:	8GiB
CPU:	Non-removable (soldered)
Boot flash:	Non-removable (soldered)

2.1.14.1 Vikings X200 builtin card reader

The internal SD card reader is known to work with GNU Boot and Linux. This means that you can boot a distribution from an SD card or a microSD card with a microSD to SD adapter. However it only supports cards that store data (and not TV Tunners, etc) and it cannot read very low level technical information about the cards (like their serial number, vendor name, etc).

2.2 Supported computer parts and peripherals

Most computer parts and peripherals don't have any compatibility issue with GNU Boot because:

- they either use some standard that is most often already implemented in the software GNU Boot reuses (storage devices like SATA drives, USB keyboards, etc),
- they are not relevant or supported for booting (for instance 3D printers, cellular network cards, etc, unless people add support for them in GNU Boot in the future). Until then they are only handled in the operating system instead (with drivers),

however there is some exceptions as some hardware is non-standard and still required for booting, these are documented in the subsections below.

2.2.1 PCI and PCIe cards with software

Some PCI or PCI express cards stores software (typically on a flash chip) that is meant to run by the boot software (typically a BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface)).

When using GNU Boot SeaBIOS images, GNU Boot will run that software regardless of if it's free or not. Most of the time this software is nonfree. The GNU Boot GRUB images won't run this software at all, so this is also one of the reasons why SeaBIOS images are reserved for more advanced users. See Section 2.4.2 [GNU Boot images types], page 22, on the various images available.

What this software does depends a lot on the card: GPUs typically contain a BIOS driver that enable to see something on the screen during boot, network cards contain code to use the card to boot the computer from the network, etc.

Sometimes this software is not useful. For instance if you don't intend to boot from the network, you don't need to run that code.

And sometimes the functionality provided by this code can even be very dangerous to use. For instance some SATA controllers contain code that can format disks with some vendor-specific format, but if you use that, since code is nonfree, if the card breaks, you end up not being able to read the data on your disks anymore unless the disk format is reimplemented with free software somehow.

Things are not always bad though, as this software can sometimes be replaced with free software like with iPXE (<https://ipxe.org>) or SGABIOS (<https://directory.fsf.org/wiki/Sgabios>).

Beside doing that there are several ways of avoiding to run that software:

- You can simply use GNU Boot GRUB images and not use GNU Boot SeaBIOS images. This will avoid running such software completely. This is the simplest solution and it also gives the strongest guarantees as users can't accidentally run such software.

- You can get computers without PCI /PCIe cards that have builtin nonfree software. All the hardware listed in the Respect Your Freedom hardware certification (<https://ryf.fsf.org/>) website should be okay. Another way would be to check the cards yourself, but a tutorial on how to do that needs to be written.
- You can backup and erase the nonfree software present in these cards with flashprog (<https://flashprog.org>) or flashrom (<https://flashrom.org>). This could be risky to do and it may require to connect a flash chip programmer to these cards and even disassemble them in some cases because flashprog or flashrom only enable to reprogram very few cards from the computer they are connected in. Here we also lack a tutorial on how to do that.
- You can modify the GNU Boot SeaBIOS images to tell it not to run this software. The Free Software Foundation tech team has a wiki. In the disable option roms with cbfstool article (<https://savannah.gnu.org/maintenance/fsf/hardware/disable-option-roms-with-cbfstool/>), they explains how to do that.

2.2.2 Supported GPUs and graphics

GNU Boot supports the GPUs that are present in the various laptops it supports with 100% free software. Some consideration apply while booting (see Section 2.4 [GNU Boot images], page 21, for more details), but so far once booted these GPU are known to works well on tested computers.

In addition for the non-laptop computers, it also supports the builtin AST graphics in the KGPE-D16 and KCMA-D8 with 100% free software, but this also comes with some limitations: in GNU/Linux it's only possible to display text but not images, so it's limited to console applications.

In the case of PCIe GPU / graphics cards, we don't know yet if it is possible to use them without running nonfree software.

If AMD, ATI, and Nvidia cards work under GNU Boot, it's because GNU Boot loaded and run the nonfree video BIOS that is present on the card.

It's possible to prevent the nonfree video BIOS from running and you can easily confirm that as the display will not work until the Linux driver is loaded.

The Free Software Foundation tech team has a wiki. In the disable option roms with cbfstool article (<https://savannah.gnu.org/maintenance/fsf/hardware/disable-option-roms-with-cbfstool/>), they explains how to do that.

And in the graphics cards article (<https://savannah.gnu.org/maintenance/fsf/hardware/graphics-cards/>) they also explain which GPU they tested.

However the Linux driver can also run nonfree software: All the current AMD, ATI, and Nvidia drivers have code to load and run (a different) initialization code provided on the card. For ATI and AMD cards the code that Linux runs is called AtomBIOS.

We don't know yet if there are cases where this code is not run (this would need to be tested by doing very simple modifications to the drivers, and the GNU Boot project also welcome help in this area).

2.2.3 Supported external card readers

GNU Boot supports some USB card readers that are viewed as mass-storage. With all that you can boot on an SD card a microSD card and it will be viewed like a mass storage USB key.

see Section 2.1 [Supported computers], page 9, inside each computer subsection to see which computers's internal card readers are supported.

2.2.4 Supported serial ports

While the serial ports are not enabled in the default GNU Boot images, GNU Boot also build debug images with the serial port enabled.

So far only the following computers have debug images with the serial port enabled:

- Asus KGPE-D16
- Lenovo ThinkPad T400
- Lenovo ThinkPad T400S
- Qemu PC (i440FX)
- Technoethical T400
- Technoethical T400s
- Vikings ASUS KGPE D16 mainboard

If you have a computer supported by GNU Boot that isn't in the list above, and that you are able to test and recover from non-working images (see Section 3.1 [Practical freedoms with flash programmers], page 27, for more details), you can ask the GNU Boot maintainers to add a debug image for your computer, and once you tested it we could update the list above.

To get the logs you can for instance get an USB<->serial adapter and a Null modem cable or adapter and after installing the picocom package, you can most of the time use the following command to get the logs:

```
picocom -b 115200 /dev/ttyUSB0
```

For USB<->Serial adapters, the serial port will most of the time be on /dev/ttyUSB0, but it can change in some configuration (when there are multiple serial ports, or with some adapters).

2.2.5 Unsupported hardware supported by projects reused by GNU Boot

The following hardware components are supported by software reused by GNU Boot, but support for them hasn't been enabled yet in GNU Boot:

- Software RAID cards: Some Silicon Image SIL3114 software RAID cards are supported by Coreboot but not enabled in GNU Boot.
- Network interfaces. Projects like iPXE has drivers for many network cards and even some Wifi cards typically used with the computers supported by GNU Boot and free distributions.

The GNU Boot project needs help to evaluate the impact of enabling these and welcome contributions in this area.

2.3 Supported operating systems

2.3.1 GNU/Linux

While GNU Boot should be able to boot almost any GNU/Linux distribution, but in some cases some configuration might be needed by the GNU Boot user. The cases that do and don't require configuration from the user will be documented in Section 2.4 [GNU Boot images], page 21, below.

Even if some cases require some configuration, GNU Boot makes sure to provide at least one way to boot free GNU/Linux distributions (see <https://www.gnu.org/distros/> for more information on these distributions) without the need to configure anything in order to make it possible for less technical users to use computers with GNU Boot, and even reinstall the GNU/Linux distribution without needing to do anything too complicated.

To make that possible, the GNU Boot contributors that proposes improvements to the project typically test GNU Boot with free distributions, and the GNU Boot project even runs automatic tests with Trisquel 11 (aramo), one of the free distributions to make sure that it can boot fine without needing any special configuration from the user.

2.3.1.1 Non-standard GNU/Linux configurations

Sometimes fully free distributions also propose experimental or non-standard configurations for very specific use cases. For instance Guix has experimental support for GNU Hurd, an experimental kernel from the GNU project, and Trisquel supports the Xen kernel, which is a virtualization solution that not supported by all GNU/Linux distributions. These configurations are not supported in the official installers of these distribution and so users are usually aware thaty they use Xen or GNU Hurd. Using GNU Boot with these configurations might require some configuration from the user. Also we would need help from users to report what works and doesn't work or what workarounds are needed to make them work with GNU Boot.

2.3.1.2 Nonfree distributions

The cases that are known not to require any configuration might also work with any GNU/Linux distributions (even the nonfree ones), however the GNU Boot project doesn't want to force contributors to download or run nonfree software to test changes, so it relies on vounteers already running such distributions to report bugs in case something doesn't work as it should.

2.3.2 Other operating systems

As for other operating systems, there is some documentation on how to boot some of them (like some BSD operating systems) on the GNU Boot website, but again we need help from vounteers already running such systems to keep the documentation up to date and inform us of what works and doesn't work.

Also if you want to do such tests, you can open a bug report on the GNU Boot bug tracker at <https://savannah.gnu.org/bugs/?group=gnuboot>.

2.4 GNU Boot images

In computers people are most familiar with, like laptops, the boot software (like the BIOS (Basic Input/Output System), UEFI (Unified Extensible Firmware Interface) or even GNU Boot) resides in the boot flash which is a data storage chip inside the mainboard (see Section 1.1.1 [boot software], page 1, for more details).

So to install GNU Boot or to update it, you will need to replace all the content of this boot flash with GNU Boot or with a newer GNU Boot version.

To do that, GNU Boot provide *image files* which are files whose content is to be written to the boot flash to overwrite everything that was previously stored by the boot flash (like previous versions of GNU Boot, the BIOS (Basic Input/Output System), etc).

These files are similar to disk images (https://en.wikipedia.org/wiki/Disk_image), ISO images (https://en.wikipedia.org/wiki/ISO_image), or ROM images (https://en.wikipedia.org/wiki/ROM_image).

We also sometime refer to the flash image files as *flash images* or simply (GNU Boot) "images" (files).

2.4.1 GNU Boot images naming

Images for specific computers can be found on the GNU Boot download area (<https://ftp.gnu.org/gnu/gnuboot/>) or in the release/roms directory if you built GNU Boot from source yourself.

For a given release (or release candidate) like GNU Boot 0.1-rc3, you can find such files inside the 'roms' directory like <https://ftp.gnu.org/gnu/gnuboot/gnuboot-0.1-rc3/roms/> for GNU Boot 0.1-rc3.

Inside you have archive files like `gnuboot-0.1-rc3_x200_8mb.tar.xz` that are specific to a specific computer (here the ThinkPad X200 with 8MiB boot flash).

see Chapter 4 [Installing or upgrading GNU Boot images], page 32, to understand how to identify which archive file correspond to which computer.

Inside each archive files, there are many smaller files that are flash images. See Section 1.1.1 [boot software], page 1, to understand what a flash image is.

The flash image files correspond to the configurations described in the Section 2.4.2 [GNU Boot images types], page 22.

So for instance if we have an image named `grub_x200_8mb_corebootfb_usqwerty.rom`, it is meant for a ThinkPad X200 with 8MiB boot flash, and it uses the GRUB software to boot, and it is configured to use a QWERTY keyboard layout.

If the image contains `seabios` in its file name instead of `grub`, it uses the SeaBIOS software to boot.

The `corebootfb` in the file name correspond to the high resolution graphics described in the previous subsection (Section 2.4.2 [GNU Boot images types], page 22).

If instead the file has `txtmode` in its name, this corresponds to the text-only low resolution that was also described in the previous subsection (Section 2.4.2 [GNU Boot images types], page 22).

2.4.2 GNU Boot images types

For a given computer, GNU Boot provides several images with different software in it. This enable the users to choose between:

- Two boot software: GRUB or SeaBIOS (BIOS (Basic Input/Output System) implementation)
- Various keyboard layouts (colemak, deqwertz, esqwerty, frazerty, frdwbepo, itqwerty, svenska, trqwerty, ukdvorak, ukqwerty, usdvorak, usqwerty).
- Low resolution or high resolution graphics.

If you are a less technical user or helping one, or don't have much time to configure things, it is a good idea to choose an image with GRUB, and a keyboard layout of your choice (the resolution is not very important, but using high resolution looks nicer) as the image with GRUB doesn't require to do any configuration in the distributions you want to boot.

Otherwise here are the advantages/disadvantages of each combinaison:

- GRUB with high resolution graphics: Images with GRUB usually don't require the user to do any configuration of the distribution. More technical users can also use that to customize the way the system boots for more security or to support unusual boot configurations (that are not typically supported by graphical installers of GNU/Linux distributions), however these more advanced configurations also come with their set of limitations.
- SeaBIOS with text-only low resolution: It implements BIOS (Basic Input/Output System) compatibility, so it is very similar to a nonfree BIOS (Basic Input/Output System) but it require users to modify some settings inside the distribution they use, otherwise the distribution still boots but usually has a black screen during the boot (which can be problematic to diagnose a non-booting distribution). The low resolution increase compatibility with various software that are typically run at boot like memtest86+ (a software that detects broken RAM chips).
- GRUB with text-only low resolution: Since these images boot with GRUB, they also don't require any configuration of the distribution and more technical users can also use them to customize the way the system boots. Compared to GRUB images with high resolution graphics:
 - the text is bigger and that there is no background picture
 - since on most supported computers, GRUB images can also load and run SeaBIOS (there is a menu entry for it), having a text-only low resolution increase the compatibility with various boot software.
- SeaBIOS with high resolution graphics:

Since these images boot with SeaBIOS they also implement some BIOS (Basic Input/Output System) compatibility, but they also require users to modify some settings inside the distribution they use. Compared with SeaBIOS images with text-only low resolution:

- they are less compatible with various boot software. This can be useful for testing if you contribute to some boot software.

- since on most supported computers, SeaBIOS images can also load and run GRUB (there is a menu entry for it when pressing the 'ESC' key at boot), having high resolution graphics can make GRUB look nicer.

2.5 SeaBIOS images requirements

2.5.1 SeaBIOS: Supported filesystems and partitions

SeaBIOS images don't need to support any filesystems because SeaBIOS is a BIOS interface implementation and BIOSes don't need to support any filesystem to load an operating system.

It supports the MBR (Master boot record) partition table.

If you use GPT (GUID Partition Table) instead, it may be possible to boot from a BIOS Boot partition instead, but it is untested. Please do report if this works for you.

Wikipedia has more information on why in its Master boot record (https://en.wikipedia.org/wiki/Master_boot_record) and BIOS Boot partition (https://en.wikipedia.org/wiki/BIOS_Boot_partition) articles.

2.5.2 SeaBIOS: Supported bootloaders

2.5.2.1 SeaBIOS: GRUB (on MBR or BIOS boot partition)

GRUB, when shipped from a GNU/Linux distribution, typically uses a graphic driver that is incompatible with GNU Boot. Because of that you typically can't see GRUB during the boot.

Even if this causes no inconvenience to you now, if there is any issues later on, you might be required to interact with GRUB due to various reasons (GRUB might stop the boot and/or you might need to interact with it to continue booting your distribution, you might need to boot in recovery mode and/or load another kernel due to various issues), and if you can't see anything you will not be able to boot anymore your distribution (unless you boot another distribution and repair it from there).

Most distribution provide a GRUB configuration in */etc/default/grub*, it looks a bit like that (it might change a bit depending on the distribution, here we used the Trisquel 11 (aramo) configuration as an example):

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'
```

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=1
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

```
# Uncomment to enable BadRAM filtering, modify to suit your needs
```

```
# This works with Linux-Libre (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

You can see it with the `sudo cat /etc/default/grub` command.

Here you will need to change `#GRUB_TERMINAL=console` to `GRUB_TERMINAL=console`.

On Trisquel can be done with the following command:

```
sudo sed 's/#GRUB_TERMINAL=console/GRUB_TERMINAL=console/' -i /etc/default/grub
```

To verify that we changed the file, we can use the following command:

```
grep '^GRUB_TERMINAL=console$' /etc/default/grub
```

It should print `GRUB_TERMINAL=console`.

We then need to run the following command to update the generated grub configuration that is in `/boot/grub/grub.cfg`:

Some distribution (here Parabola) have a slightly different configuration file in `/etc/default/grub`.

Instead of having this:

```
# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console
```

We have:

```
# Uncomment to use basic console
GRUB_TERMINAL_INPUT=console

# Uncomment to disable graphical terminal
#GRUB_TERMINAL_OUTPUT=console
```

So we instead need to use this command:

```
sudo sed 's/#GRUB_TERMINAL_OUTPUT=console/GRUB_TERMINAL_OUTPUT=console/' \
```

```
-i /etc/default/grub
```

To verify that we changed the file, we then need to use the following command:

```
grep '^GRUB_TERMINAL_OUTPUT=console$' /etc/default/grub
```

It should then print *GRUB_TERMINAL_OUTPUT=console*.

We then need to run the following command to update the generated grub configuration that is in */boot/grub/grub.cfg*:

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

If you use Guix system instead, you don't have a configuration file like that. So you will need to modify the bootloader section of your system configuration to add (*terminal-outputs '(console)*) inside your bootloader section. It should look a bit like that:

```
(bootloader (bootloader-configuration
  (bootloader grub-bootloader)
  (targets '(file-system-label "Guix_image"))
  (terminal-outputs '(console))))
```

Then to update the GRUB configuration, you will need to reconfigure your system:

```
sudo guix system reconfigure /etc/config.scm
```

Since most distribution hides the GRUB menu by default unless they detects some issue, if one day you need to interact with GRUB (for instance to boot with an older kernel), at boot, you will need to first wait for SeaBIOS to load.

It should look like that:

```
SeaBIOS (version rel-1.14.0-27-g64f37cc5)
```

Press ESC for boot menu.

At this point, do not press *ESC* and wait a bit until it starts. You should see something like that:

Booting from Hard Disk...

You can then press the *Escape* or *ESC* key multiple times until it interrupts GRUB, right after SeaBIOS. At this point you should see something more or less like that (instead of a completely black screen):

```
GNU GRUB version 2.12
+-----+
|*Trisquel GNU/Linux                               |
| Advanced options for Trisquel GNU/Linux           |
| Trisquel GNU/Linux 11.0.1, Aramo (11.0.1) (on /d→|
| Advanced options for Trisquel GNU/Linux 11.0.1, →|
+-----+

Use the ↑ and ↓ keys to select which
entry is highlighted.
Press enter to boot the selected OS, `e'
to edit the commands before booting or
`c' for a command-line.
```

2.6 GRUB images requirements

2.6.1 GRUB: Supported configuration files

2.6.1.1 grub.cfg

When booting, the GNU Boot GRUB image selects the first entry, which is *Load Operating System (incl. fully encrypted disks)*.

It has limited heuristics to find LVM, raid and encrypted partitions.

Once it finds a filesystem, it tries to find the `gnuboot-grub.cfg`, `osboot-grub.cfg`, `autoboot-grub.cfg`, `coreboot-grub.cfg`, `grub.cfg`, in that order, in the `boot`, `grub`, `grub2`, `boot/grub`, or `boot/grub2` directories.

So if it loads the wrong distribution, you can put grub commands that load the right distribution in `/boot/gnuboot-grub.cfg` in the partition it loads the wrong distribution from.

2.6.2 GRUB: Supported partitions

2.6.2.1 GRUB: GPT (GUID Partition Table) and MBR (Master boot record)

The GRUB images support GPT and MBR partitions.

2.6.2.2 GRUB: LVM2

While basic usage of LVM2 is supported and well tested, it also has various backends like for RAID0, RAID5, etc, and these are not well tested and might not all be supported.

2.6.2.3 GRUB: GNU/Linux RAID

GNU Boot GRUB images support some RAID combinations, but we also lack information on which ones do and don't work. Please contact us if you have any information on that.

Due to limitations in the grub configuration of the GNU Boot GRUB images, we only support 10 RAID volumes (`md/0` to `md/9`), help is welcome to fix that.

2.6.2.4 GRUB: RAID formats of RAID card vendors

While some GNU/Linux distributions support that, this is completely untested in GNU Boot and it is probably not supported. In addition we don't support these RAID cards yet (See Section 2.2.5 [Unsupported hardware supported by projects reused by GNU Boot], page 19). Help is welcome for identifying if we can add support for these.

2.6.2.5 other partition types

Right now, GNU Boot probably supports more partition types. We are considering removing them in GNU Boot.

Would they be missed if we remove them from GNU Boot? If so please contact us.

3 Flash programmers

3.1 Practical freedoms with flash programmers

Getting a compatible flash programmer is not strictly required to use GNU Boot.

While you need a compatible flash programmer to upgrade from the nonfree BIOS to GNU Boot on some computers, it is possible to avoid using one by either getting a compatible laptop with an alternative boot software (like Canoeboot, Coreboot, GNU Boot, Libreboot, etc) already installed, or by having someone do the installation for you (for instance in an install party, or by paying a company to do a "Installation Service" of an alternative boot software).

Note that these distributions are very different: they do not support the same computers, and when they do, supporting them can require nonfree software (making it undesirable to support these computers in GNU Boot), even for computers supported by 100% free distributions like Canoeboot.

See Section 1.4 [Other free boot software distributions], page 5, and more specifically see Section 1.4.3 [How much free software are other distributions], page 7, for more details on the freedom implications of using other distributions than GNU Boot and what it means for hardware support.

However if you know how to disassemble computers supported by GNU Boot, or that you can get help to do that, it is very strongly advised to get a compatible flash programmer: with it you will be able to recover from non-booting computers, and this will enable you to do a lot more things.

Not only you will be able to test and run previously untested GNU Boot images, but you will also more easily be able to customize the GNU Boot images that you use. Since GNU Boot is slowly switching to Guix to build the software it reuses, and that Guix has lots of ways to customize software, over time this will also enable you to customize GNU Boot even more to better suit your needs.

3.2 Setup and components needed to use a flash programmer

To reprogram the boot flash of a laptop (for instance the Lenovo ThinkPad X200), you will need several thing:

- A flash chip programmer
- A clip or test probe to connect to the boot flash.
- Some wires (usually female<->female jump wires) to connect the flash chip programmer to the clip.
- An extra computer running GNU/Linux with an USB port that works. Any computers should work, however if you care about security, since this extra computer will have the ability to modify the GNU Boot image that it is reading or writing, you need to be able to trust this extra computer, so choose this extra computer accordingly.

To reprogram the boot flash of a desktop, server or workstation mainboard (for instance the Asus KGPE-D16), you will need several thing:

- A flash chip programmer

- A way to extract the boot flash from the mainboard. This can be done by hand but if you do that a lot, at the end the pins of the boot flash might break. If that happens, new boot flash are relatively easy to find. If you want to avoid breaking the chip instead, you can either use specialized tools to extract the flash chip (like a scalpel, some tweezers for stamps that you can try to put between the chip and the socket to remove the chip without bending its pins, or a chip extractor), or you can glue something on top of the boot flash for easier extraction.
- If your programmer doesn't have a socket where to plug the mainboard flash chip you will need a breadboard or a boot flash socket, and a way to connect the breadboard to the flash programmer (for instance female<->female jump wires and some pin rows).

3.3 Flash chip package

The flash chip use various standard physical and pin layout.

The supported laptops have either a boot flash in the SOIC-8, SOIC-16 or WSON-8 package.

The desktop, servers and workstations mainboard typically use a flash chip in the DIP-8 package.

3.4 Known issues with flash programmers

3.4.1 Wire length

If the length of the wires connecting the programmer to the chip is too long, then there will be signal integrity issues and flashprog or flashrom might not find the boot flash at all.

If that happens, the best fix is to use shorter wires. It is also sometimes possible to workaround by lowering the speed at which the programmer talks to the boot flash.

3.4.2 Programmer SPI speed

On some flash chip programmers, the speed at which it access the flash chip can be configured. In this case using the lowest speed possible can sometimes make a given flash programmer work when there are signal integrity issues (for instance when the wires connecting the programmer to the boot flash are too long). How to do that is documented in the flashrom and flashprog manuals, inside the programmer sections. Such options have names like spispeed, or divisor, depending on the programmer being used.

Note that some programmers don't have the ability to lower their speed.

3.4.3 Lenovo ThinkPad X200 with the MX25L6405D (8MiB)

For some reasons the signal integrity is bad with the MX25L6405D (8MiB) boot flash on the Lenovo ThinkPad X200. Because of that it is advised to have the shortest cables possibles (see Section 3.4.1 [Wire length], page 28, for more details) but also to check the compatibility of various flash programmers with this boot flash on this laptop. On programmers that are known to work, you might still need to lower the SPI speed (see Section 3.4.2 [Programmer SPI speed], page 28, for more details) to be able to detect, read or write the boot flash.

3.5 Supported flash programmers

3.5.1 OpenMoko Neo1973 Debug board (V2+)

The Neo1973 Debug board (V2+) from OpenMoko can be used but it will require to solder some pin headers on the programmer itself.

If you want to use this programmer to reprogram a laptop boot flash, you will need to get Female/Female Jumper Wires and a test clip corresponding to your boot flash (like the Pomona 5252 test clip for SOIC-16 or the Pomona 5250 test clip for SOIC-8).

If instead you want to reprogram a DIP-8 boot flash, you will need to get Female/Female Jumper Wires, some PIN headers and a breadboard.

Boot flash	Works?
DIP-8 SPI	Yes
Lenovo ThinkPad X200 with MX25L6405D	No

3.5.2 Vikings CH341a USB Programmer

The CH341a USB Programmer from Vikings can be used without having to solder anything.

If you want to use this programmer to reprogram a laptop boot flash, you will need to get Female/Female Jumper Wires (like the "Jumper Wire F/F, 20pcs" from Vikings) and a test clip corresponding to your flash chip (like the "Pomona 5252 IC TEST CLIP SOIC 16 (2 X 8)" or "Pomona 5250 IC TEST CLIP SOIC 8 (2 X 4)" from Vikings).

If instead you want to reprogram a DIP-8 boot flash, nothing special is needed as this programmer has a builtin socket for them.

Boot flash	Works?
DIP-8 SPI	Yes
Lenovo ThinkPad X200 with MX25L6405D	No

3.5.3 Zerocat Chipflasher v2 and its variations

The Chipflasher v2 is shipped with a builtin free firmware that is compatible with flashrom and that can be replaced: Zerocat provides documentation that explains you how to do that and alternative firmwares do exist.

Zerocat also provides documentation that seems sufficient to build a Chipflasher yourself and they even sells kits, though we didn't test building one ourselves.

The builtin free firmware is capable of controlling various hardware components to adapt to a wide variety of electrical / electronic conditions typically found when programming laptops boot flash.

Building your own slightly different version the Chipflasher and/or installing another firmware that is completely different has real impacts on both the compatibility with laptops boot flash and the way of using the Chipflasher.

All these differences can be useful, otherwise the Chipflasher v2 official documentation on the Zerocat website or the chipflasher git repository would not document all these in depth.

In order to support well the Zerocat Chipflasher v2 and potentially its various variations as well, we chose to take into account these differences by considering Zerocat Chipflasher

v2 with hardware and software variations as flash programmers distinct from the Zerocat Chipflasher v2.

So for instance a Zerocat Chipflasher v2 without any builtin firmware, with a different firmware than the one that came with the Chipflasher v2 (like kick2-connect instead of kick2-flashrom), or a Zerocat Chipflasher DevKit (for instance that you built yourself) will be considered as distinct flash programmers.

3.5.3.1 Zerocat Chipflasher v2

The Chipflasher v2 from Zerocat can be used without having to solder anything. It is shipped with a Pomona 5252 IC TEST CLIP for SOIC 16, an adapter for DIP-8 flash chips and various cables.

Since the Zerocat Chipflasher v2 has a builtin (free) firmware, using that makes things much more easy (for instance it is compatible with a wide variety of USB serial port adapters), but in some situations that can have security implications (as the firmware can for instance be modified while shipping the flasher).

For the security implications, see Chapter 6 [Security], page 39, for more background on security first, and then for the security related to using flash chip programmers without or with builtin firmwares, see Section 6.4.3 [Installing with a 100% hardware flash chip programmer], page 45, and Section 6.4.4 [Installing with a flash chip programmer that has a firmware], page 45.

Boot flash	Works?
Lenovo ThinkPad X200 with MX25L6405D	Yes

3.5.3.2 Zerocat Chipflasher v2 without builtin firmware

The Chipflasher v2 without builtin firmware from Zerocat can be used without having to solder anything. It is shipped with a Pomona 5252 IC TEST CLIP for SOIC 16, an adapter for DIP-8 flash chips and various cables.

It is also shipped with a firmware by default. This firmware is located in a flash chip that is inside a DIP-8 socket. This means that you can remove this flash chip.

If you are not used to do removing chips from DIP-8 sockets, you might consider using some specialized tools to help (like a scalpel, some tweezers for stamps that you can try to put between the chip and the socket to remove the chip without bending its pins, or a chip extractor).

While doing that, you might also need to protect the Chipflasher v2 from static electricity, as this tend reduce the lifespan of electronics. You could for instance by using an antistatic wrist strap. Alternatives like touching some electrical ground also exist but you need to know what you are doing.

Once done you will still need to load a firmware into the Chipflasher, but the firmware will come straight from the computer that will control the Chipflasher.

While this avoids security issues, since there is no builtin firmware anymore, you will need to build and load this firmware yourself. The official Chipflasher v2 documentation from Zerocat explains how to do that. Ideally you will also need to find a way to verify the signatures of the source code of this firmware. And finally loading the fimrware may not work with all serial port adapters, though compatability has already been improved over time.

For the security implications, see Chapter 6 [Security], page 39, for more background on security first, and then for the security related to using flash chip programmers without or with builtin firmwares, see Section 6.4.3 [Installing with a 100% hardware flash chip programmer], page 45, and Section 6.4.4 [Installing with a flash chip programmer that has a firmware], page 45.

4 Installing or upgrading GNU Boot images

GNU Boot provides flash images for specific computers that can be found on the GNU Boot download area (<https://ftp.gnu.org/gnu/gnuboot/>).

But depending on your threat model, it could be a good idea to build GNU Boot from source yourself instead, to avoid certain security attacks. See Chapter 6 [Security], page 39, section for more context with security and threat models and Chapter 8 [Building GNU Boot from source], page 48, for more details about the security attacks mentioned above.

Once GNU Boot is downloaded or built, you will need to understand which files you need to install or upgrade. See Chapter 2 [Supported hardware and configurations], page 9, chapter for more details on how to do that.

4.1 Installation and upgrade instructions

The GNU Boot manual doesn't have well integrated installation or upgrade instructions yet but some generic installation and upgrade instructions can be found in the GNU Boot website. We need help to migrate these instructions in the manual and make them easier to understand.

5 Using GNU Boot

5.1 Using GNU Boot with QEMU

The GNU Boot project also release images for QEMU. They can be useful for trying GNU Boot, especially the GRUB images, to see how to boot various distributions. If you contribute to GNU Boot, they can also be useful to run quick tests tests.

However note that testing with these QEMU images don't fully replace tests on real computers. For instance a distribution or operating system might work on QEMU but not work on real hardware due to an incomplete graphic driver for the real hardware GPU.

5.1.1 Quick test with QEMU

If you just want to try an image to see how it looks like, after installing QEMU, you can use the following command:

```
qemu-system-i386 -M pc \
-bios grub_qemu-pc_2mb_corebootfb_usqwerty.rom
```

Here you need to replace *grub_qemu-pc_2mb_corebootfb_usqwerty.rom* by the path to the image you want to try.

If you don't have *qemu-system-i386* you can either ask users of your distribution for help, to understand which package you need to install, or use *qemu-system-x86_64* instead.

Alternatively, the *qemu* package from Guix also provides both (*qemu-system-i386* and *qemu-system-x86_64*).

5.1.2 Booting a distribution with QEMU

In the previous section (Section 5.1.1 [Quick test with QEMU], page 33) we didn't boot a distribution.

To boot an x86 32bit distribution, you can use the following (QEMU) command:

```
qemu-system-i386 \
-bios grub_qemu-pc_2mb_corebootfb_usqwerty.rom \
-M pc \
-m 128M \
-cpu kvm32 \
-enable-kvm \
-blockdev \
'{"driver":"file","filename":"rootfs-i686.img","node-name":"libvirt-1-storage"}' \
-blockdev \
'{"node-name":"libvirt-1-format","driver":"raw","file":"libvirt-1-storage"}' \
-device \
'{"driver":"virtio-scsi-pci","id":"scsi0","bus":"pci.0","addr":"0x4"}' \
-device \
'{"driver":"ahci","id":"sata0","bus":"pci.0","addr":"0x5"}' \
-device \
'{"driver":"ide-hd","bus":"sata0.0","drive":"libvirt-1-format","id":"sata0-0-0"}'
```

If it fails with the following error:

```
qemu-system-i386: failed to initialize kvm: Permission denied
```

You can fix that by removing the `-enable-kvm` from the command above. It will always work but it will make booting the virtual machine slower.

Alternatively, you can try to fix this by adding the proper permissions to `/dev/kvm`. To do that you can either consult your distribution documentation / wiki or ask other users of that distribution for help. Also, this will only work if your processor supports KVM (most computer do).

In addition, for the command to work you will also need to provide the *rootfs-i686.img* file (otherwise it will fail with *Could not open 'rootfs-i686.img': No such file or directory*).

If you don't know how to do that, you can install Guix first (it can be installed on top of another GNU/Linux distribution) and generate it with Guix.

You can consult either the GNU Boot build instructions (<https://www.gnu.org/software/gnuboot/web/docs/build/>) or the Section “Installation” in *GNU Guix reference manual* for how to install Guix.

Then for distributions (like PureOS or Trisquel) that provide Guix versions older than 1.4.0, or if you don't know the Guix version you have, you will need to update Guix with the following commands:

```
guix pull
```

It could take a long time though (maybe 1 hour or more depending on the speed of the computer and network).

Then make sure you are using the latest Guix with the following command (this needs to be done each time you use a new shell and want to use the latest Guix when Guix is installed on top of another distribution):


```
hash guix
```

Then you can copy the following inside the `guix-i686-linux.scm` file:

```
;;; This file is free software; you can redistribute it and/or modify it
;;; under the terms of the GNU General Public License as published by
;;; the Free Software Foundation; either version 3 of the License, or (at
;;; your option) any later version.
;;;
;;; This file is distributed in the hope that it will be useful, but
;;; WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
;;; GNU General Public License for more details.
;;;
;;; You should have received a copy of the GNU General Public License
;;; along with this file. If not, see <http://www.gnu.org/licenses/>.
(use-modules (gnu) (gnu bootloader) (gnu packages linux))
(operating-system
  (host-name "guix-i686-linux")
  (kernel linux-libre-lts)
  (bootloader (bootloader-configuration
    (bootloader grub-bootloader)
    (targets '(file-system-label "Guix_image"))
    (terminal-outputs '(console))))
  (file-systems (append (list (file-system
    (device (file-system-label "Guix_image"))
    (mount-point "/")
    (type "ext4"))))
    %base-file-systems)))
```

And finally you can generate the `rootfs-i686.img` file with the following command:

```
install -m 644 \
  "$(guix time-machine \
    --commit=8e2f32cee982d42a79e53fc1e9aa7b8ff0514714 \
    -- \
    system image --system=i686-linux guix-i686-linux.scm)" \
  rootfs-i686.img
```

This will create the `rootfs-i686.img` that you need in the current directory.

You can then boot it with the QEMU command we gave before. You should then see this:

```
This is the GNU system. Welcome.
guix-i686-linux login: _
```

You then can login as the `root` user with no password (by typing `root` and then pressing the `ENTER` key), and once done you can shut it down with the `shutdown` command.

To use this with another distribution or image, you will have to find a way to download or generate the `rootfs-i686.img` file that contains the distribution that you want to test yourself and adapt the QEMU command accordingly.

See Section 5.1.3 [Booting an x86 64bit distribution with QEMU], page 37, for an example with a different image.

5.1.3 Booting an x86 64bit distribution with QEMU

In the previous subsection (Section 5.1.2 [Booting a distribution with QEMU], page 34) we booted a 32bit distribution as it works fast on both 32bit and 64bit x86 computers and it uses less RAM.

In this example we need 256 MiB of RAM (It is set in the command below with *-m 256M*).

So to boot an x86 64bit distribution, you can use the following command:

```
qemu-system-x86_64 \
-bios grub_qemu-pc_2mb_corebootfb_usqwerty.rom \
-M pc \
-m 256M \
-cpu kvm64 \
-enable-kvm \
-blockdev \
'{"driver":"file","filename":"rootfs-x86_64.img","node-name":"libvirt-1-storage"}' \
-blockdev \
'{"node-name":"libvirt-1-format","driver":"raw","file":"libvirt-1-storage"}' \
-device \
'{"driver":"virtio-scsi-pci","id":"scsi0","bus":"pci.0","addr":"0x4"}' \
-device \
'{"driver":"ahci","id":"sata0","bus":"pci.0","addr":"0x5"}' \
-device \
'{"driver":"ide-hd","bus":"sata0.0","drive":"libvirt-1-format","id":"sata0-0-0"}'
```

Here too, if it fails with the following error:

```
qemu-system-x86_64: failed to initialize kvm: Permission denied
```

You can also fix that by removing the *-enable-kvm* from the command above. It will always work but it will make booting the virtual machine slower. see Section 5.1.2 [Booting a distribution with QEMU], page 34, for pointers on how to make *-enable-kvm* work to not make the virtual machine boot slower.

Here you will need to provide the *rootfs-x86_64.img* file.

If you don't know how to do that, like mentioned in the previous section (Section 5.1.2 [Booting a distribution with QEMU], page 34), you can install Guix and update it if necessary.

Once done don't forget to run this command to use the latest Guix:

```
hash guix
```

You can then copy the following inside the `guix-x86_64-linux.scm` file:

```
;;; This file is free software; you can redistribute it and/or modify it
;;; under the terms of the GNU General Public License as published by
;;; the Free Software Foundation; either version 3 of the License, or (at
;;; your option) any later version.
;;;
;;; This file is distributed in the hope that it will be useful, but
;;; WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
;;; GNU General Public License for more details.
;;;
;;; You should have received a copy of the GNU General Public License
;;; along with this file. If not, see <http://www.gnu.org/licenses/>.
(use-modules (gnu) (gnu bootloader) (gnu packages linux))
(operating-system
  (host-name "guix-x86_64-linux")
  (kernel linux-libre)
  (bootloader (bootloader-configuration
    (bootloader grub-bootloader)
    (targets '(file-system-label "Guix_image"))
    (terminal-outputs '(console))))
  (file-systems (append (list (file-system
    (device (file-system-label "Guix_image"))
    (mount-point "/")
    (type "ext4"))))
    %base-file-systems)))
```

And finally you can generate the `rootfs-x86_64.img` file with the following command:

```
install -m 644 \
  "$(guix time-machine \
    --commit=8e2f32cee982d42a79e53fc1e9aa7b8ff0514714 \
    -- \
    system image --system=x86_64-linux guix-x86_64-linux.scm)" \
  rootfs-x86_64.img
```

Once booted, you should then see this:

```
This is the GNU system. Welcome.
guix-x86_64-linux login: _
```

Here too, you can login as the `root` user with no password (see Section 5.1.2 [Booting a distribution with QEMU], page 34, for more details), and you can also shut it down with the `shutdown` command.

5.1.4 Booting more distributions with QEMU

6 Security

6.1 Very basic introduction on security

This section gives some basic and generic introduction to computer security and risk management with free software in mind. It is a pre-requisite to understand the next sections in the Security chapter.

If you are already familiar with threat modelling, that you know how to choose software with security criteria in mind, but also with other criteria in mind as well, then you can skip this section and jump directly to the sections after this one that are directly related to GNU Boot.

If not, note that very basic good practices like running only free software that comes from the distribution(s) you use, applying security updates and doing regular backups can help avoiding most of the issues most of the time because with free software, a lot of the security is non-intrusive and simply work without requiring the user to do anything.

However sometimes this is not the case, so if you want to understand if you need to take extra steps to limit some risks you may find non-acceptable for your use cases, you can read at least this "Very basic introduction on security" section.

Unfortunately most people don't know these things, and this most likely includes many members of the free software community as well.

And without the basics here, seriously talking about computer security and risk management would be meaningless, misleading, or would be restricted to some very narrow topic that isn't necessarily relevant to most people.

6.1.1 Threat modelling or what security really is about

Security is a process. To really make it work you need to understand various threats and if or how to respond to them (this is called *threat modelling*). This depends on your life, your use cases, etc. It boils down to what is important to you and what is not.

For instance if a Company that makes nonfree software lose money, this is probably problematic for them but it is likely not your problem, and it could even be a good thing, depending on the details.

In contrast, if your computer is stolen, you could lose your data, and the person or entity that stole it may have access to all your files. You also might have a hard time getting another equivalent computer. Here the damage can depend a lot on how you live your life, on who / what entity stole the computer, or even if your computer has access to any important data at all, or if you can actually be in a situation where you can relatively easy get computer.

All these are security threats but they don't threaten the same people or organisations in the same ways.

In addition, sometimes there are security features like "encryption with LUKS", or "hard disk encryption" that can be promoted, that protects against some known attack (like someone or some entity stealing your laptop to access your data), but not everything in security can be classified as features and even features are part of bigger processes that respond to specific threats.

For instance if you have "hard disk encryption" but that all your data are backed up in some other computers you don't trust and that you misunderstood the risks of doing that, then this could be really catastrophic.

In addition some things are hard to fit into features. "Good code quality", or "I can trust the people doing this software because I know them well" or "I don't want to use this software because the people that do it know me personally and I feel uncomfortable because of that even if I trust them" are not things that a given software project can easily advertize.

Not everything is technical and sometimes it can boil down to things like "this project is well known, looks serious and Wikipedia also agrees on that", so it looks safer to use than "another project that advertize 100% security" because "according to my friends, 100% security doesn't exist" or that "the security features of this software create other security issues that affect you more than what it's supposed to fix".

Also note that in general some security features also have downsides, such as making it harder to use the computer, making it harder to fix issues, etc, so not everybody might want these security features.

For instance "disk encryption" when it is really secure, makes the computer slower when copying very big files, makes recovering from data corruption harder, and you even need to remember a passphrase (that you could forget and so lose access to your data).

Making sure that nobody else gets the passphrase is hard (you need to not be seen, recorded, filmed, etc when you type it, it needs to be random enough and long enough not to be guessed, etc). And finally all that can complicate backups as well depending on if you also want the backups to be encrypted as well or not.

So not everybody wants to encrypt their disk, this is why GNU/Linux installers give users the choice of doing it or not doing it.

Security is also not everything, for instance you might care more if a given software has features that you want, or if it fits you better, or has better long term availability guarantees or better support than a more secure one, etc.

A big part of threat modelling is also, when you have things to protect, to also understand who or what to protect these things from.

For instance a state doesn't have the same capabilities, goals, or even risk than an individual. So there are things states are willing to do that individuals are not and vice-versa. Attacks can also have costs for the attacker, so that also limit their use to situation where the gain outweighs the costs. But that is only useful if you first understand what is important for you.

6.1.2 Impact of malware

People running 100% free software distributions usually don't end up with malware on their computers if they apply (security) updates.

But this can happen to other people that run other GNU/Linux distributions with (very) bad luck and bad practices.

If someone that runs GNU/Linux installs random software from places like Aur or Pip that don't do any checks on the software being provided through them, that software is sometimes malware, though this is quite rare and Aur or Pip typically communicate on such incidents and the specialized press talks about it.

Places like Aur or Pip also provide nonfree software, and this is by design, so people running 100% free software distributions try to avoid such places anyway.

The consequences of installing malware is quite bad as the computer cannot be trusted anymore and the advertised solution is usually to reinstall completely the GNU/Linux distribution, which is relatively easy to do.

This however brings additional problems that are much more complex to deal with:

- When restoring backups, people need to make sure they don't contain code that, once restored, will re-infect the computer. Restoring a backup of before the infection should be safe, but sometimes malware take time (months) before being detected. And making sure that backups made after the infection are safe is too complex for most computer users.
- Such advice (reinstalling GNU/Linux) also assumes that the malware didn't infect the computer firmwares or boot software. Usually there are also many security features in places to make that a lot harder but they don't always work, especially with software that is installed from untrusted sources when such sources don't limit what the software can do through security mechanisms.

6.2 Security considerations when using GNU Boot

If GNU Boot is already installed, Someone else than you that has physical access to the computer could Boot GNU/Linux and use flashrom to replace GNU Boot with a malicious modified version.

If you run software that you don't trust that have or can get root permissions, it also has the ability to replace GNU Boot with a malicious modified version.

If these things happens you will need to reinstall both GNU/Linux and GNU Boot. When doing that you will need not to run the potentially compromised GNU/Linux and GNU Boot installations that are on the computer.

How to prevent such attacks needs to be added later on in this manual.

If the attacker has access to significant resources, with the same attack vectors (through untrusted software or by booting GNU/Linux and using flashrom), it can also replace the firmware of various peripherals like keyboards, storage devices (SSDs, USB keys, etc) with malicious versions that can take control of the computer, but to do that it needs to dedicate or have dedicated resources to the specific families of chip present in these peripherals.

Untrusted software can also come from cards that can be plugged in the computer, like GPUs or express-cards for laptops, so the same consideration apply for these.

In the case of laptops, it would also be possible to replace the firmware that controls the keyboard and various other power management features. This is really problematic. See Section 1.3.1 [How much free software is GNU Boot?], page 3, for more detail on the issue.

These kind of attacks is not specific to GNU Boot though, however some GNU Boot configurations can be used to protect from some of these attacks. How to do that will need to be added later on in this manual. See Section 6.2.1 [Secure boot and restricted boot], page 42, for more details.

6.2.1 Secure boot and restricted boot

A very common security feature typically found in other boot software that is part of many computers is called *secure boot*.

When it cannot be turned off, it becomes an anti-feature and the Free Software Foundation (<https://www.fsf.org/>) calls it *restricted boot*.

In 2012, the Free Software Foundation (<https://www.fsf.org/>) wrote a whitepaper (<https://www.fsf.org/campaigns/secure-boot-vs-restricted-boot/campaigns/secure-boot-vs-restricted-boot/whitepaper.pdf>), on the topic and advised that:

The best solution currently available for operating system distributions includes:

1. fully supporting user-generated keys, including providing tools and full documentation for booting and installing both modified and official versions of the distribution using this method;
2. using a GPLv3-covered bootloader to help protect users against the dangers of Restricted Boot;
3. avoiding requiring or encouraging users to trust Microsoft or any company which makes proprietary software; and
4. joining the FSF and the broader free software movement in pressuring computer distributors to facilitate easy and independent installation of free software operating systems on any computer.

GNU Boot supports various security mechanism: GRUB is a GPLv3-covered bootloader that GNU Boot reuses, and it supports user-generated keys or other security mechanism that that don't require any signing keys.

GNU Boot also obviously doesn't Trust keys from companies that make proprietary software.

At the end when used correctly, the security features provided by GNU Boot thanks to the software it reuses (like GRUB) can provide similar or stronger security guarantees than the UEFI secure boot with different security features that you may or may not want to use depending on your threat model.

The GNU Boot Website contains various information on how to use such security features, but they are also documented in the *GNU GRUB manual* as well in more details. Since the GRUB version GNU Boot uses might be older than the online GRUB manual, you can use Guix to install the manual of older GRUB versions (see *GNU Guix reference manual* for more details).

All the security mechanism described in the GRUB manual or GNU Boot website are compatible with users freedom.

6.3 Security considerations when updating GNU Boot

All the consideration in Section 6.2 [Security considerations when using GNU Boot], page 41, apply.

In addition when updating GNU Boot you will need to decide if you want to trust binary images provided by the GNU Boot project.

If you do, it is a good idea to verify the signature to make sure that what you are about to install really comes from the GNU Boot maintainers. If you choose not to trust these binary images you will need to build GNU Boot from source.

See Chapter 8 [Building GNU Boot from source], page 48, for information that can help decide if (not) trusting the binaries is best for your or not.

When you do that it is also a good idea to verify that the source code you use is signed by the GNU Boot maintainers (See Section 8.5 [Authenticating the GNU Boot source code], page 51).

To do that you will need to install Guix. And here too it is a good idea to make sure that the installation method you use also checks signatures. For instance guix packages provided by distributions are signed, but the Guix installation script from the Guix project isn't.

You can consult either the GNU Boot build instructions (<https://www.gnu.org/software/gnuboot/web/docs/build/>) or the Section “Installation” in *GNU Guix reference manual* for how to install Guix.

6.3.1 Checking the signatures of GNU Boot images

To verify the signatures of an image, you first need to copy the following inside the *neox.asc* file:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mDMEZm2YmxYJKwYBBAHaRw8BAQdAWDcG3nLKPosKY1A7DABOYkWN3B2Hq4lhrIJx
NnlQVieOKEfkcml1biAnbmVveCcgQm91cm1hdWx0IDxuZW94QGEtbGVjLm9yZz6I
lgQTFggAPhYhB0I8JqXe7sX6nN3Ve1e8JqNocRb2BQJmbZzfAhsDBQkDwmcABQsJ
CAcCBhUKCQgLAgQWAgMBAh4BAheAAAJEF8JqNocRb2248BA0IQBGj4/2doph2Q
sfYLyQ0zMR8EaX5P9C2zmDZiaoXBVAQDsh1dvMsLfHpkRdYbRqo6uTU4QTlml5q0H
Cz2s0QyWC7QmQWRyaWVuICduZW94JyBCb3VybwF1bHQgPG51b3hAZ251Lm9yZz6I
mQQTfGgAQIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgBYhB0I8JqXe
7sX6nN3Ve1e8JqNocRb2BQJmbZzsAhkBAAJEF8JqNocRb2D+IA/iY1SD/npaG8
z1DIjJr7XUREFCB6fELSEYRONKGT+wmGAP0dLdP4xG70anliz+76ik/pndHRZdof
qKQKROEBXv/vCrgzBGZtmJsWCSsGAQQB2kcPAQEHDZQd7U/DRPK5/qk35dzeG5d
pnS/0FesbRrgZTSMHEsviH4EGBYIACYWIQTiPCa13u7F+pzd1XpXvCajaHEW9gUC
Zm2YmwIbIAUJA8JnAAAKCRBXvCajaHEW9o68AP9VQXiDp5BWXSE6N2rz2+b/Dpck
Ho0MMww2awmBvfU6xgD+OpkDEgo/in+m5Pcb0Xa8IqDC1fZZbGsLKEfBSiZhawG4
OARmbZibEgorBgEEAZdVAQUBAQda7nIe/ImvNJ8WZbA+ZcvxrwLbXLwu3XuCPi0c
aIJw9zoDAQGHiH4EGBYIACYWIQTiPCa13u7F+pzd1XpXvCajaHEW9gUCZm2YmwIb
DAUJA8JnAAAKCRBXvCajaHEW9gZBAQDJFP0bkwf3j3YuZROGGEsgwATfohbh8GYB
y35gwjcaigD/fjDVjNnxjppqGFOLvEkSMZgjNpejar69Msf1sRH8tGAg=
=Zqad
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

You will then need to import the key into the GPG keyring. This can be done with the following command:

```
gpg --import neox.asc
```

Once done you will be able to verify images signed by neox, one of the two GNU Boot maintainers who until now signed all the releases.

To do that you will first need to download the images and their signature. For QEMU

- gnuboot-0.1-rc6_debug_qemu-pc_2mb.tar.xz
- gnuboot-0.1-rc6_debug_qemu-pc_2mb.tar.xz.sig

```
wget https://ftp.gnu.org/gnu/gnuboot/gnuboot-0.1-rc6/roms/gnuboot-0.1-rc6_debug_qemu-p
wget https://ftp.gnu.org/gnu/gnuboot/gnuboot-0.1-rc6/roms/gnuboot-0.1-rc6_debug_qemu-p
gpg --verify gnuboot-0.1-rc6_debug_qemu-pc_2mb.tar.xz.sig
gpg: assuming signed data in 'gnuboot-0.1-rc6_debug_qemu-pc_2mb.tar.xz'
gpg: Signature made Mon 24 Mar 2025 10:54:59 AM CET
gpg:          using EDDSA key E23C26A5DEEE5FA9CDD57A57BC26A3687116F6
gpg: Good signature from "Adrien 'neox' Bourmault <neox@gnu.org>" [unknown]
gpg:          aka "Adrien 'neox' Bourmault <neox@a-lec.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: E23C 26A5 DEEE C5FA 9CDD  D57A 57BC 26A3 6871 16F6
```

6.4 Security considerations when installing GNU Boot

If you are installing GNU Boot, you could be in one of the situations described in the following subsections.

All these need different security considerations.

6.4.1 Installing from a computer with a nonfree BIOS

To install GNU Boot on a given computer, you will simply run some software that does the installation for you.

This software will replace the nonfree BIOS that is on the boot flash (See Section 1.1.1 [boot software], page 1, for more information on what is a boot flash).

The main issue with this approach is that the existing BIOS could be malicious and because of that there is no way to know if your computer was already compromised at the time of the installation.

In practice it means that you will need another computer and a flash chip programmer to either check that the GNU Boot image you installed was not modified and/or to simply reinstall GNU Boot.

Once done, you will also need to not boot anymore your previous GNU/Linux installation and reinstall GNU/Linux as well. See Section 6.1.2 [Impact of malware], page 40, for more details on the difficulties of doing that properly.

6.4.2 Installing from a computer with another boot software distribution

The security consideration here will depend a lot on the exact image you used. If it is fully free and that the computer was never compromised, the same consideration than Section 6.3 [Security considerations when updating GNU Boot], page 42, do apply.

Else it is mostly similar to Section 6.4.1 [Installing from a computer with a nonfree BIOS], page 44.

See Section 1.4.3 [How much free software are other distributions], page 7, for information on the freedom of various boot software distributions. In addition, if the distribution you

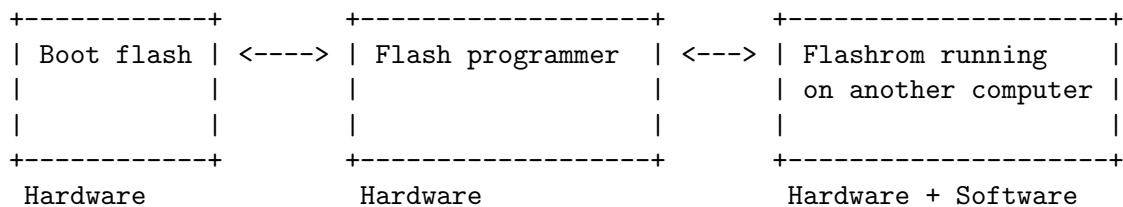
use is not listed there it most likely means that it either ships nonfree software and/or that it has no commitment to only include free software.

6.4.3 Installing with a 100% hardware flash chip programmer

To do the installation you will need:

- A computer to install GNU Boot on.
- A 100% hardware flash chip programmer
- Another computer that runs the flashrom program

The setup should look more or less like that:



The previous subsections already explain how to deal with the computer that you are about to install GNU Boot on. For instance they explain the risk with the firmware that controls the keyboard on laptops.

They also explain the risk of running untrusted code regardless of if it comes from the Boot flash or the operating system that is installed on the laptop.

The flash programmer being only hardware doesn't need special considerations.

So the main concern here is to make sure that the computer running flashrom is trustworthy.

How to bootstrap this trust will need to be added later on in this manual, but it boils down to adapting what is known as a key ceremony (especially how it was done for Let's Encrypt) to GNU Boot.

In addition, if the boot flash or operating system of the computer that you install GNU Boot on is compromised, overwriting its content with a new GNU Boot image will get rid of the compromise.

And if the operating system of that computer is also compromised, you will need to first (re)install GNU Boot, and then make sure reinstall the operating system without ever running the code from the previous installation. A way to do that could be to erase the disk on another computer, for instance with the *wipe* command which comes from the *wipe* package in Guix.

6.4.4 Installing with a flash chip programmer that has a firmware

Compared to Section 6.4.3 [Installing with a 100% hardware flash chip programmer], page 45, here we have one more complication: we need to trust the code that runs inside the flash chip programmer.

In the case of the Zerocat Chipflasher v2 (See Section 3.5.3.1 [Zerocat Chipflasher v2], page 30, for more details on this flash chip programmer), the builtin firmware is on a flash chip that uses the I2C protocol. This chip can simply be removed and the firmware can be loaded through the serial port instead. If you do that you then are in the same case than

Section 6.4.3 [Installing with a 100% hardware flash chip programmer], page 45, which is way more simple.

In other cases, like with a flash chip programmer made with an Arduino, things are more complex.

The `frser-duino` (<https://github.com/urjaman/frser-duino>) firmware is free and it can easily be (re)built and installed into an Arduino to make it become a flashrom compatible flash chip programmer. Note that some Arduinos do require level shifter to be able to safely program the boot flash found on the computers supported by GNU Boot.

However if you build a flash chip programmer with an Arduino, you also implicitly trust the bootloader of the Arduino you use. If you don't want to trust this bootloader, you will either need to completely get rid of the bootloader or build and install one yourself.

Both solutions do require to use an FTDI chip to program the arduino processor directly without needing to run the bootloader code. This is for instance described in the FTDI Breakout with additional ISP connector (<https://arduino.stackexchange.com/questions/30564/ftdi-breakout-with-additional-isp-connector>) thread on stackoverflow. On an Arduino duemillanove this requires to solder 1 connector and to use 4 female->female jump wire cables and to use the avrdude software.

It is also possible to use another Arduino and software to do these, but then it only shift the issue to the other Arduino that also has a bootloader that you might not want to trust.

6.5 Trusting hardware

Like with software, someone else than you that has physical access to hardware (like the computer you use, your flash chip programmer, etc) can also modify the hardware to do malicious things. An well known example is hardware keyloggers.

This can often be detected with visual inspection of the circuits. Making modified chips is also possible and probably relatively cheap with various chip packaging processes and technologies (probably around 10 000 Euros to assemble different components in a single chip, this is commonly done for RAM and CPU / System on a chip for instance).

Another possible workaround would be to try to get hardware in ways that don't enable an attacker to target you and/or communities you belong to in advance, for instance by not ordering online, etc.

Once done you could then install only free software on this hardware, and then use various temper-evident ways to seal it (and that also make sure that somebody else cannot do the exact same seal). Along with other software access control, this is also a good way to make sure that the software on the laptop isn't modified by someone else.

And it also enable you to modify everything you wish: if you do that you probably know it, or write about it, and you then need to re-do the seals. How to do all that will need to be explained later on in this manual.

7 Maintenance and repairs

7.1 Replacing the boot flash on a desktop, server or workstation

If you break the boot flash (for instance by breaking its pins), it can easily be replaced.

To do that you need to use a boot flash of the same chip package (see Section 3.3 [Flash chip package], page 28, to understand what is a chip package) and storage size and that uses the SPI protocol (currently all supported computers use SPI boot flash).

To check the compatibility of the chip with Flashrom you can use the following command:

```
flashrom -L
```

And here's the command if you use flashprog instead:

```
flashprog -L
```

In the output we can see for instance a 2MiB boot flash that are known to work well:

```
flashrom v1.3.0 on Linux 6.11.11-gnu (x86_64)
flashrom is free software, get the source code at https://flashrom.org
```

```
Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Supported flash chips (total: 576):
```

Vendor	Device	Test	Known	Size	Type
		OK	Broken	[kB]	

```
(P = PROBE, R = READ, E = ERASE, W = WRITE, - = N/A)
```

```
[...]
```

Spansion	S25FL016A	PREW		2048	SPI
----------	-----------	------	--	------	-----

```
[...]
```

Alternatively, if you are able to find one, you can also use the same boot flash model than the broken one or any of the boot flash that were shipped with the computer model (see Section 2.1 [Supported computers], page 9, for information on that).

You might also need a flash programmer and another computer to put GNU Boot on the new boot flash (see Chapter 3 [Flash programmers], page 27, for more details on flash chip programmers).

8 Building GNU Boot from source

Currently building GNU Boot flash images on two different computers will produce slightly different images.

This is a problem as it prevents people from easily verifying that the official flash images really correspond to the source code published by GNU Boot, and having the ability for anyone to verify that increases the security guarantees.

The Reproducible builds (<https://reproducible-builds.org>) project helps publicizing this problem and helps distributions and software to fix it.

So while GNU Boot also started working to fix this problem the work just stated and isn't complete yet, so in the meantime if you care about this type of risks, it might be a good idea to build GNU Boot from source yourself.

The GNU Boot website has instructions for building GNU Boot at the following URL: <https://www.gnu.org/software/gnuboot/docs/build/>.

See Section 8.5 [Authenticating the GNU Boot source code], page 51, as GNU Boot has ways to prevent network attacks from tempering with the source code you are downloading.

8.1 Supported distributions for building GNU Boot binaries

GNU Boot is currently based on the latest version of Libreboot that doesn't ship nonfree software, and it also uses an older version of Coreboot to support certain computers that are not supported anymore in Coreboot. Because of that the versions of various software that GNU Boot builds are old and cannot be built anymore on recent distributions.

While there is work to fix that by both updating that software to more recent versions and to also allow to build older versions on newer distributions, in the meantime we need to workaroud this issue by using specific distributions to build GNU Boot.

People managed to build GNU Boot with the following distributions:

- PureOS 10 (byzantium)
- Trisquel 10 (nabia)
- Trisquel 11 (aramo)

And these cannot build GNU Boot yet:

- Guix: The issue is documented in the Bug#66188 (<https://savannah.gnu.org/bugs/index.php?66188>)

Since GNU Boot 0.1 RC6, the GNU Boot status page (<https://www.gnu.org/software/gnuboot/status>) started including information on which distribution was used to build a given release ("This release was built [...] on Trisquel Aramo (11.0.1) [...]").

Using the same distribution and version of the distribution lowers the risk of ending with a computer that doesn't boot anymore (for fixing that you will most likely need to disassemble your computer and use another computer and a flash programmer to recover it).

Also note that you don't use PureOS 10 (byzantium) or Trisquel 10 (nabia), there are many ways to run them on top of other GNU/Linux distributions.

If you run Guix (either as an operating system or on top of another distribution), Parabola, Trisquel 10 (nabia), Trisquel 11 (aramo), you can use `debootstrap` to create a chroot of Trisquel 11 (aramo) or PureOS 10 (byzantium). Here are the packages you need to install depending on your distribution:

Host distro	Chroot distro	Required packages
Guix	PureOS 10 (byzantium)	<code>debootstrap</code>
Guix	Trisquel 10 or 11	<code>debootstrap</code>
Parabola	PureOS 10 (byzantium)	<code>debootstrap</code> , <code>pureos-archive-keyring</code>
Parabola	Trisquel 10 (nabia)	<code>debootstrap</code> , <code>trisquel-keyring</code>
Trisquel >= 10	Trisquel 10 (nabia)	<code>debootstrap</code> , <code>trisquel-keyring</code>

Once you have a chroot, you can either configure it and chroot inside or convert it to run inside container engines like LXC, LXD, Docker (with `debuerreotype` if your distribution has a package for it), etc.

It is also possible to install Trisquel 10 (nabia) or PureOS in a virtual machine. Note that PureOS doesn't sign its releases so we copied the official PureOS checksums found in several subdirectories in <https://downloads.puri.sm/byzantium/resources/distros/pureos/20230614/> in the GNU Boot repository. The commits of GNU Boot are usually signed by its maintainers, so it's also possible to have a full chain of trust.

PureOS also has docker images on Docker Hub, and it also has one for PureOS byzantium (<https://hub.docker.com/r/pureos/byzantium>). On Docker Hub, The PureOS images made by Puri.sm are the only images that follow the Free Distro Guidelines (<https://www.gnu.org/distros/>). Also note that it is not possible to easily check the integrity of images coming from docker hub so by using them you blindly trust Docker Hub. The only way to check the images is to create your own image and compare it with the one hosted on docker hub.

8.2 Installing Guix

While GNU Boot doesn't build yet on top of Guix, it started using some Guix packages to build part of GNU Boot. While this provides many benefits, you will need to install Guix on top of a supported distribution to build GNU Boot binaries.

There are many ways to install Guix, and they are well documented in the *GNU Guix manual* especially in the Section "Installation" in *GNU Guix manual*.

If you're in a hurry you can simply run the following command (it will also take care of enabling some substitutes):

```
sudo apt install guix
```

However, while building GNU Boot is supposed to work with out of the box once Guix is installed, users are advised to update the Guix daemon as explained in the Section "Upgrading Guix" in *GNU Guix manual* to avoid any security issues.

8.2.1 Enabling Guix substitutes

Once Guix is installed, It is very strongly advised to to *enable substitutes*.

If you don't, Guix will build everything from source. This has many implications.

First, the build time will most likely take more than one day. This is because unlike other distributions, Guix bootstraps compilers to improve freedom and security. And doing

that result in extremely long build times. See Section “The Full-Source Bootstrap” in *the GNU Guix manual* for more details.

Then, packages that built in the past might not build anymore (see the Adventures on the quest for long-term reproducible deployment (<http://guix.gnu.org/en/blog/2024/adventures-on-the-quest-for-long-term-reproducible-deployment/>) blog post for more details and how to workaround the issue.

Finally, if you don’t have enough RAM, or not enough RAM per CPU cores, building with Guix can also fail (see Section 8.2.3 [GNU Boot configuration for Guix], page 51, for how to workaround this issue).

If the instructions to install Guix you followed didn’t mention how to enable substitutes, you can still find documentation on it in the Section “Substitutes” in *GNU Guix manual*.

8.2.2 Enabling Guix all substitutes servers

Recent versions of Guix have both the *ci.guix.gnu.org* and *bordeaux.guix.gnu.org* substitute server enabled, while older versions only have the former. GNU Boot needs to have both servers enabled.

If you don’t enable both, the build could be really slow (see Section 8.2.1 [Enabling Guix substitutes], page 49, for more details).

To check if you also have *bordeaux.guix.gnu.org*, you can use the following command:

```
sudo grep -q \  
    '#7D602902D3A2DBB83F8A0FB98602A754C5493B0B778C8D1DD4E0F41DE14DE34F#' \  
    /etc/guix/acl && echo "bordeaux.guix.gnu.org enabled"
```

If it prints *bordeaux.guix.gnu.org enabled* then you have *bordeaux.guix.gnu.org* enabled. If not you need to enable it.

To do that you first need to create a file with the *bordeaux.guix.gnu.org*.pub public key (and trust this manual that this is really the right public key):

```
cat > bordeaux.guix.gnu.org.pub << EOF  
(public-key (ecc (curve Ed25519)  
(q #7D602902D3A2DBB83F8A0FB98602A754C5493B0B778C8D1DD4E0F41DE14DE34F#)))  
EOF
```

And to then authorize the *bordeaux.guix.gnu.org* public key:

```
sudo guix archive --authorize < bordeaux.guix.gnu.org.pub
```

Alternatively, if you do not want to blindly trust the command above (which comes from this manual and not from Guix), you could simply update Guix and get the file from Guix instead, but updating Guix the first time can be quite long to do.

To do that first update Guix:

```
guix pull
```

Then authorize the *bordeaux.guix.gnu.org* public key that comes from Guix:

```
sudo guix archive --authorize < \  
    ~/.config/guix/current/share/guix/bordeaux.guix.gnu.org.pub
```


8.2.3 GNU Boot configuration for Guix

At the time of writing, Guix can use about 2GiB of RAM per core for updates. Building packages can also use some RAM but the types of packages that GNU Boot will build are unlikely to require that much RAM per core.

Even with substitutes enabled the build can still fail due to the lack of RAM, because GNU Boot builds packages that are not in Guix.

If you are affected, it is possible to limit the amount of RAM used during the build by limiting the number of cores used by Guix by passing `-with-guix-build-cores=1` to the GNU boot `./configure` script. This will pass the `'-c 1'` and `'-M 1'` options to guix build.

8.2.4 Managing space with Guix

Guix keeps the files it downloads or builds (in `/gnu/store`) in order to speed up things, but if you use Guix extensively, at some point it might use too much storage space.

Guix users are able to decide when to free up space by running the `'guix gc'` command manually, but they can also control what to remove with various criteria. The Section “Invoking guix gc” in *GNU Guix manual* has more details on how to do that.

8.3 Building from tarballs and/or offline

At the moment, building GNU Boot from a tarballs is unsupported, so you will have to build GNU boot from a git repository.

GNU Boot also releases git bundles, so you might also be able to build GNU Boot with them, but even with that, you will still need to download source code from the Internet.

In addition, some instructions need to be written to do that, and we also need to make sure that the result is doesn't differ too much from images built with a repository downloaded with git.

8.4 Downloading the GNU Boot source code with git

To download GNU Boot you first need to have git installed. To do that you can run the following command:

```
$ sudo apt install git
```

You can then download the GNU Boot source code with the following command:

```
$ git clone https://git.savannah.gnu.org/git/gnuboot.git
```

8.5 Authenticating the GNU Boot source code

Once the GNU Boot source code has been downloaded, it might be a good idea to check if it wasn't modified.

This way if for some reasons the source code was modified, it will be detected. This can also protect other people as checking also improves the probability of catching active attacks. If you're unsure if it's worth the effort, See Chapter 6 [Security], page 39, for an introduction to security and threat modelling.

If you want to do that, once you downloaded the GNU Boot source code (see Section 8.4 [Downloading the GNU Boot source code with git], page 51, if you missed it) and installed Guix, you then need to go inside the GNU Boot source code:

```
$ cd gnuboot
```

And you can then authenticate the source code with the following guix command:

```
$ guix git authenticate \
d4223088cbfe8a347626638d32902ba2323b25 \
"E23C 26A5 DEEE C5FA 9CDD D57A 57BC 26A3 6871 16F6" \
-k origin/keyring
```

Depending on the Guix version, it could print something like that (for recent versions):

```
guix git: successfully authenticated commit d4223088cbfe8a347626638d32902ba2323b25
```

or something like that (if you just installed guix from Trisquel 11 (aramo)):

```
Authenticating commits d4223 to 4a77965 (241 new commits)...
```

See Section “Invoking guix git authenticate” in *GNU Guix manual* or the Authenticate your Git checkouts! Guix blog post (<https://guix.gnu.org/en/blog/2024/authenticate-your-git-checkouts/>) for more details.

The question that remains is then how to make sure that "E23C 26A5 DEEE C5FA 9CDD D57A 57BC 26A3 6871 16F6" is the right key.

To do that the GnuPG software can help (see *its manual* for now to use it if you are interested) but the solution to this problem is not technical but social and could require significant time and effort.

To solve this problem you will need to build some sort of chain of trust between you and the person who controls the "E23C 26A5 DEEE C5FA 9CDD D57A 57BC 26A3 6871 16F6" key (here Adrien 'neox' Bourmault) with or without the help of the GnuPG software.

Wikipedia has a bit more information on the problem in its Web of trust (https://en.wikipedia.org/wiki/Web_of_trust) article, and the The GNU Privacy Handbook (<https://www.gnupg.org/gph>) has a section about Building your web of trust (<https://www.gnupg.org/gph/en/manual.html#AEN554>), that contains advises on how to do that, especially in the part about "Key validation".

8.6 Initializing the GNU Build system

GNU Boot uses the GNU Build system (also known as autotools). Compared to other build systems, this build system is very flexible, and it is way more familiar and easier to use for end users.

It requires the installation of some dependencies. You can install them with the following command:

```
sudo apt install autoconf libtool make
```

It requires you to generate some files. To do that you first need to make sure you are in the gnuboot directory. If you did `cd gnuboot` before in the same terminal, then everything is fine, if not, you need to run the `cd gnuboot` commands.

Then you can then need to run the following command to generate the required files:

```
./autogen.sh
```

Once done you can proceed to the Section 8.8 [Installing dependencies for building GNU Boot], page 54, section.

8.7 Configuring the GNU Boot build

Once the GNU Build system is initialized (see Section 8.6 [Initializing the GNU Build system], page 52, for how to do that), we can then use it to install the dependencies required for building GNU Boot.

To do that you need to first run the configure script. This can be done with the following command:

```
./configure
```

It will check if everything is fine. When it happens the output will look more or less like that:

```
[...]
config.status: creating Makefile
config.status: creating manual/Makefile
config.status: creating resources/packages/i945-thinkpads-install-utilities/Makefile
config.status: executing depfiles commands
config.status: executing libtool commands
Configuration options:
  GUIX_BUILD_MAX_CORES: not limited.
  KVM: enabled
  build_cpu: x86_64
```

If it didn't detect any errors or that you don't need to change the way things are built (for instance to use less resources or workaround the lack of some CPU features), you can simply continue to the next section (Section 8.8 [Installing dependencies for building GNU Boot], page 54).

If it detects any errors or that you want or need to limit the amounts of resources GNU Boot uses during the build, you might need to change the build configuration.

The full list of options is available with the following command:

```
./configure --help
```

However this also contains many options that are not unused by GNU Boot yet. The following options are however taken into account by GNU Boot:

- `-disable-kvm`
- `-with-guix-build-max-cores=N`
- `-with-xz-extra-args="ARG ARG [...]"`

They are sometimes mentioned in the rest of this manual. To learn what they do you can also use the command mentioned above and try to find the options above in the command output:

```
./configure --help
```

In the case of the `-disable-kvm` option, there is also some documentation on the underlying problem in Section 5.1.2 [Bootting a distribution with QEMU], page 34. Note that the options is named almost in the same way but it affects two different programs (GNU Boot and QEMU).

Though we end up with exactly the same issue in QEMU and GNU Boot because GNU Boot uses QEMU during some tests if you use a 64bit distribution to build GNU Boot.

In the case of `-with-guix-build-max-cores=N`, there is also some documentation in the Section 8.2 [Installing Guix], page 49.

8.8 Installing dependencies for building GNU Boot

Once the GNU Build system is initialized and configured (see Section 8.6 [Initializing the GNU Build system], page 52, and see Section 8.7 [Configuring the GNU Boot build], page 53, for how to do that), you can finally proceed to install the required dependencies.

Once you are in the `gnuboot` directory, you can simply run the following command to do that:

```
sudo make install-dependencies
```

This will take care of detecting the distribution you use and installing the dependencies from your distribution. See Section 8.1 [Supported distributions for building GNU Boot binaries], page 48, for a list of distributions on which this is supposed to work.

If for some reasons it fails on a supported distribution, this is a bug and this should be reported to GNU Boot either on the `gnuboot-patches` mailing list (<https://lists.gnu.org/mailman/listinfo/gnuboot-patches>) or on Savannah (<https://savannah.gnu.org/bugs/?group=gnuboot>).

8.9 Building a GNU Boot release

Once the GNU Build system is initialized, configured and that the build dependencies are also installed (see Section 8.6 [Initializing the GNU Build system], page 52, Section 8.7 [Configuring the GNU Boot build], page 53, and Section 8.8 [Installing dependencies for building GNU Boot], page 54, for how to do that), you can finally proceed to building the same files than with a GNU Boot release.

To do that you can run the following command:

```
make release
```

It will take many hours as it builds images for all the supported computers, and it also build the website, the manual, etc, exactly like for a GNU Boot release.

If for any reasons, the build is interrupted, you will need to delete both the `bin/` and `bin-dbg/` directories if they exist, otherwise the build system will assume that the build went fine and this might create issues later on, especially if you want to build a release.

Once done it will create many files which are described below:

- `release/roms/`: Inside this directory, each archive contains the images for a supported computer. See Section 2.1 [Supported computers], page 9, and Section 2.4 [GNU Boot images], page 21, for more details.
- `release/roms-dbg/`: Inside this directory, each archive contains the images for a supported computer. See Section 2.2.4 [Supported serial ports], page 19, for more details on debug images. The `resources/coreboot/README.debug` file inside the GNU Boot source code details the difference between debug images and regular images.
- `release/i945-thinkpads-install/`: This directory has tools (and their corresponding source code) required for installing GNU Boot on some computers without the need of a flash programmer, however we don't have good documentation yet on how to use these. Only certain computers are compatible.

- release/gnuboot-[_..._]_src.tar.xz: Source code of projects reused by GNU Boot (Coreboot, GRUB, etc).
- release/gnuboot-website-[_..._]_tar.gz: GNU Boot website (HTML files, GNU Boot manual).
- release/gnuboot-source-[_..._]_bundle: Complete GNU Boot source code.
- release/lbssg-[_..._]_bundle: Source code for the static website generator used to build the GNU Boot website.

If you instead just want to build an image for your computer, see instead the instructions for building GNU Boot binaries (<https://www.gnu.org/software/gnuboot/docs/build/>) on the GNU Boot website.

Also note that these instructions also depend on some of the instructions above in this manual.

9 Helping GNU Boot

The GNU Boot project needs help with this manual, specifically on moving information from the GNU Boot website to this manual.

In general there is also a lot of ways to help the GNU Boot project (from reviewing website pages for very simple mistakes or outdated information, testing GNU Boot images, etc).

See the Helping GNU Boot (<https://www.gnu.org/software/gnuboot/git.html>) page on the GNU Boot website for the areas where we need help and on how to help practically speaking (how to contact the project, where to send bug reports, etc).

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

- be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
 - C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
 - D. Preserve all the copyright notices of the Document.
 - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
 - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
 - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
 - H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/licenses/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept index

B		S	
BIOS (Basic Input/Output System)	1	secure boot	42
boot flash	1		
boot software	1	T	
		threat modelling	39
D			
DIP (Dual in-line package)	28	U	
		UEFI (Unified Extensible Firmware Interface)....	1
F			
flash chip	1	W	
flash images	21	WSON (Very-very-thin small-outline no-lead	
flash memory	1	package)	28
I			
image files	21		