

Bootable applications

An introduction

Repo ↗



[github.com/michaelo/
bootable-applications](https://github.com/michaelo/bootable-applications)

Vibe check

Practical info

Repo ↗

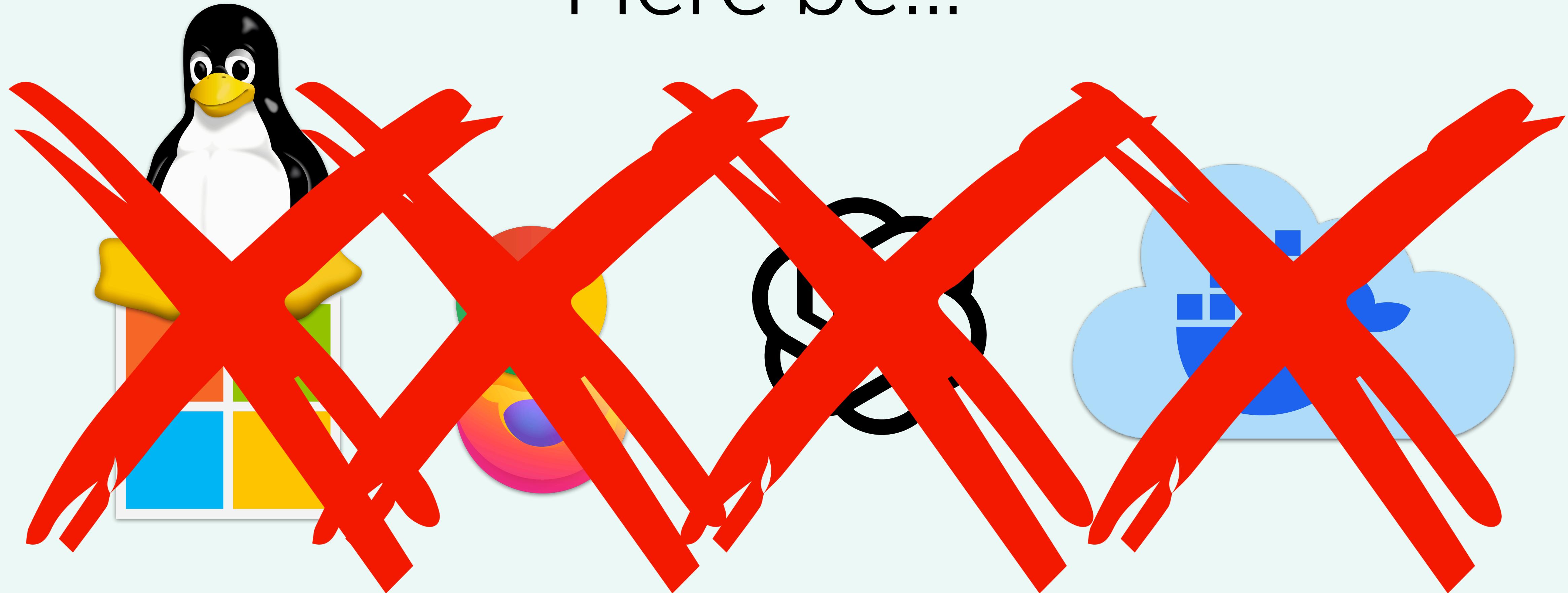


[github.com/michaelo/
bootable-applications](https://github.com/michaelo/bootable-applications)

Scope

But... why!?

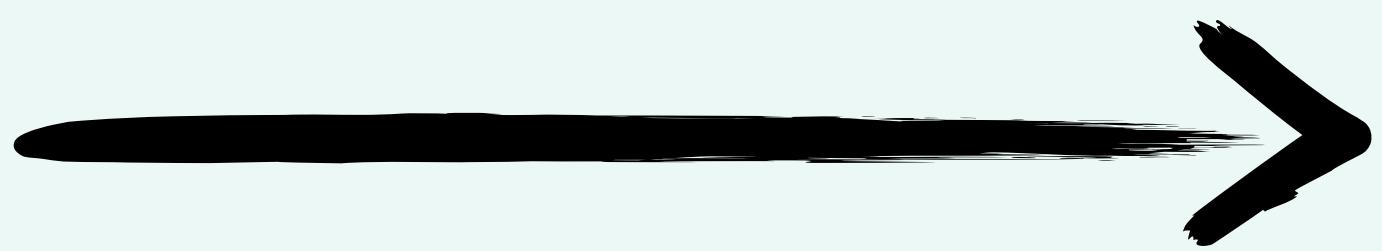
Here be...



... but there might be dragons
in the code



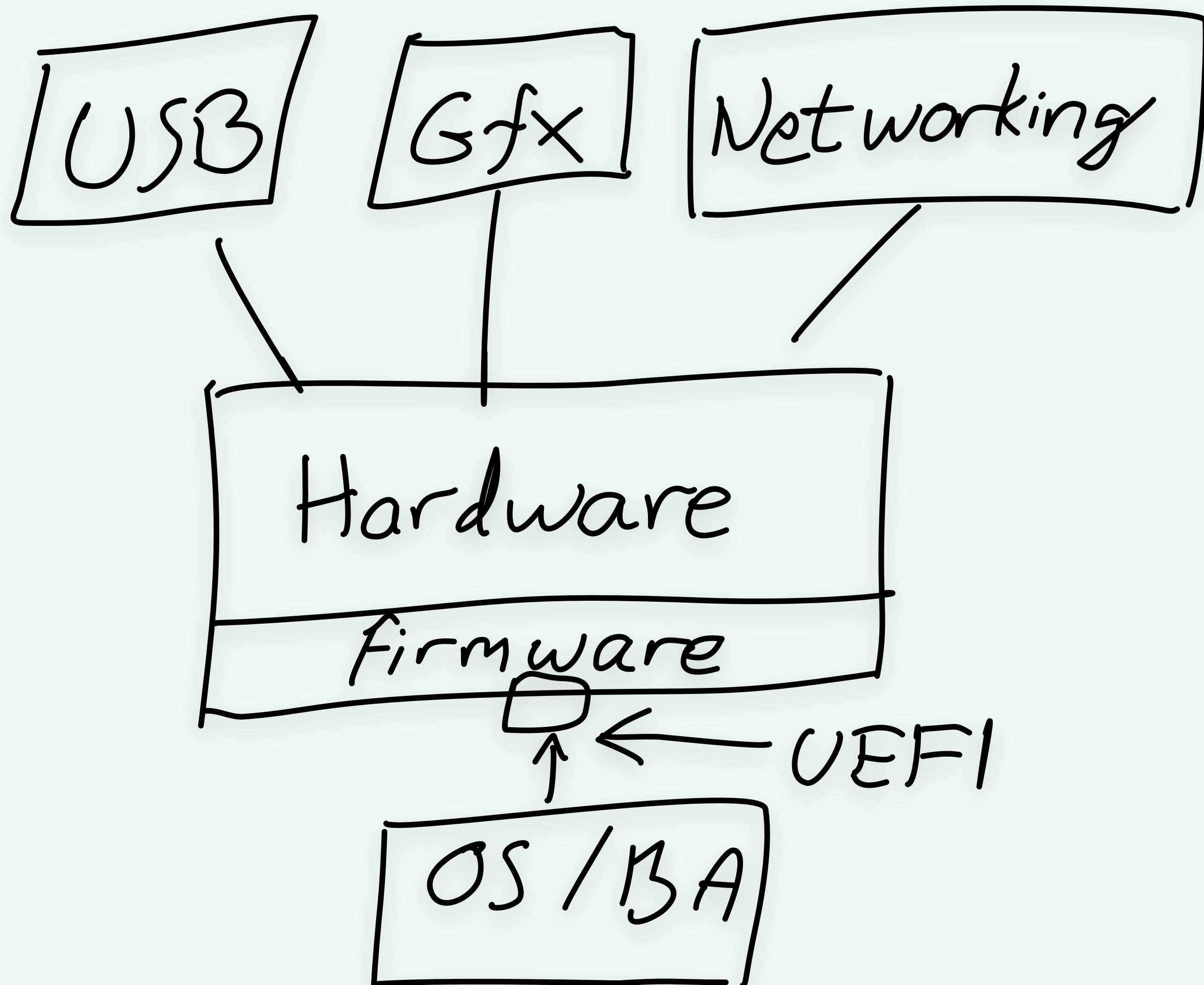
UEFI?



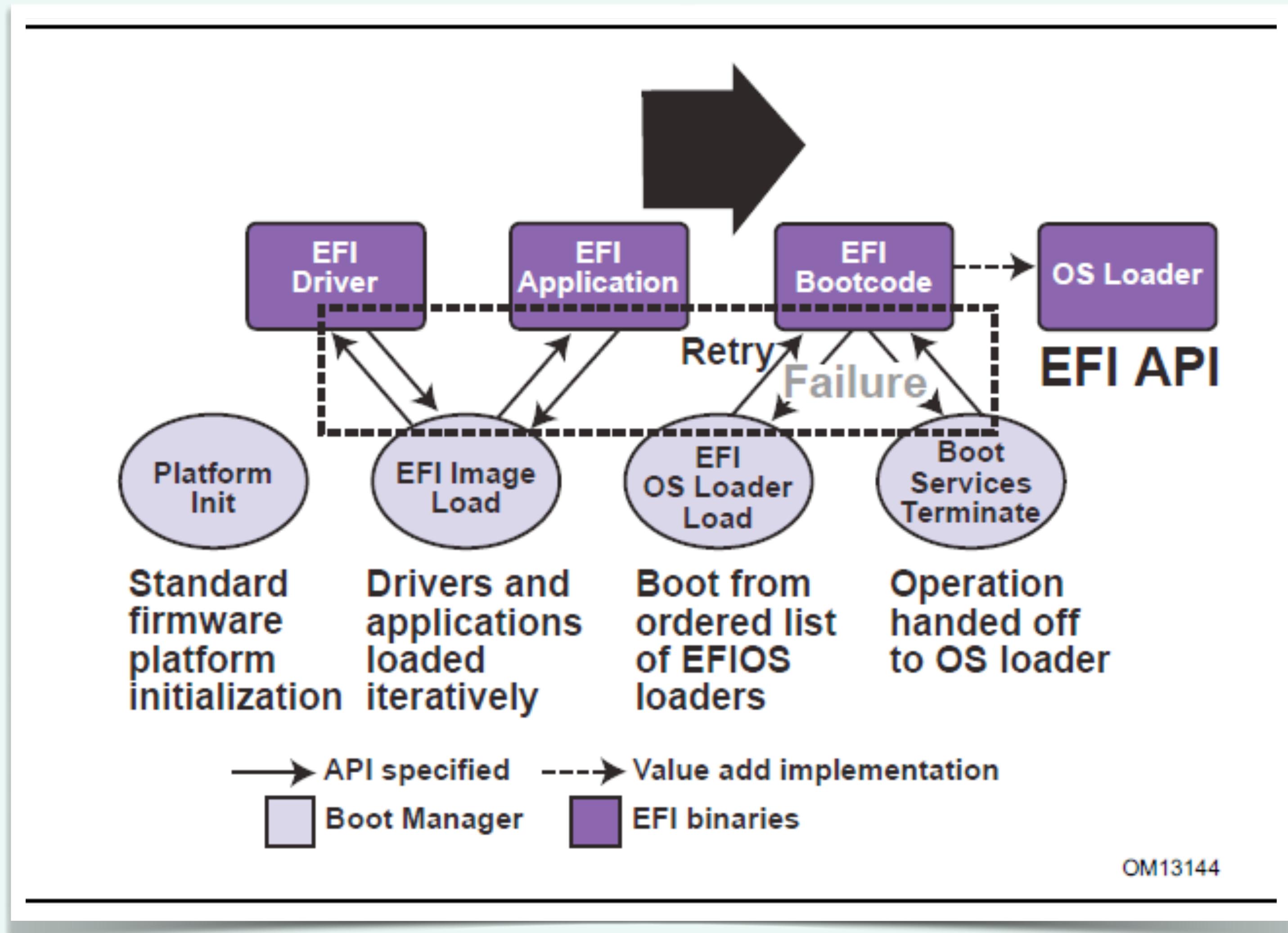


Unified Extensible Firmware Interface

… Intel - Microsoft - ARM - AMD - Apple …
… IBM - nvidia - Cisco - Broadcom …



Well defined specification FTW



<https://uefi.org/specs/UEFI/2.10/>

4. EFI System Table

4.1. UEFI Image Entry Point

4.1.1. EFI_IMAGE_ENTRY_POINT

4.2. EFI Table Header

4.3. EFI System Table

4.4. EFI Boot Services Table

4.5. EFI Runtime Services Table

4.6. EFI Configuration Table & Properties Table

4.7. Image Entry Point Examples

5. GUID Partition Table (GPT) Disk Layout

6. Block Translation Table (BTT) Layout

7. Services – Boot Services

8. Services – Runtime Services

9. Protocols - EFI Loaded Image

10. Protocols – Device Path Protocol

11. Protocols – UEFI Driver Model

12. Protocols – Console Support

13. Protocols – Media Access

14. Protocols – PCI Bus Support

15. Protocols – SCSI Driver Models and Bus Support

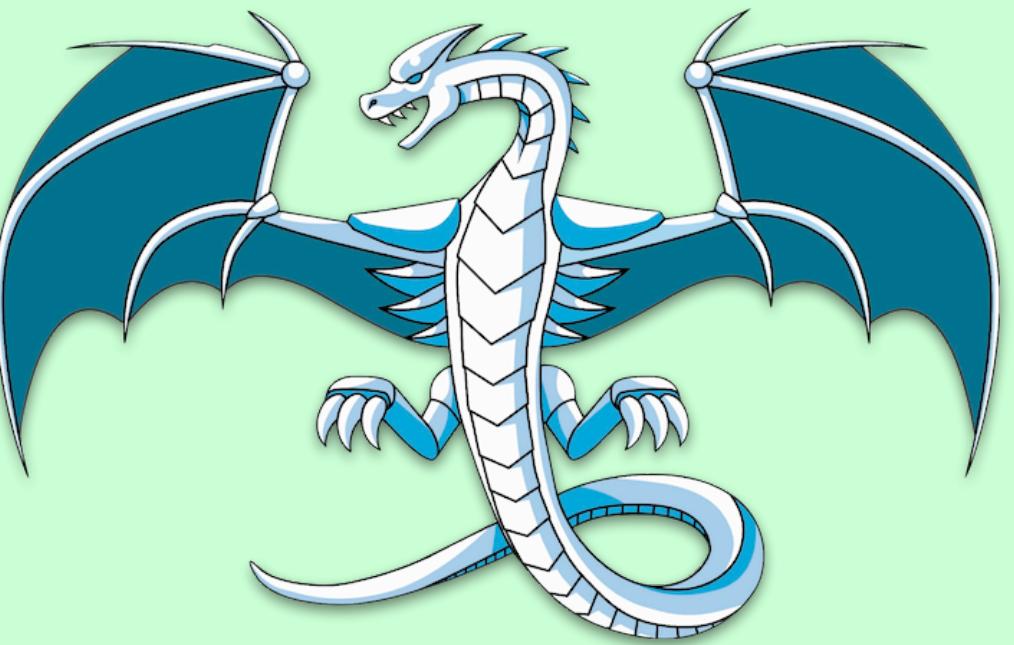
16. Protocols – iSCSI Boot

17. Protocols – USB Support

18. Protocols – Debugger Support

19. Protocols – Compression Algorithm Specification

The tools



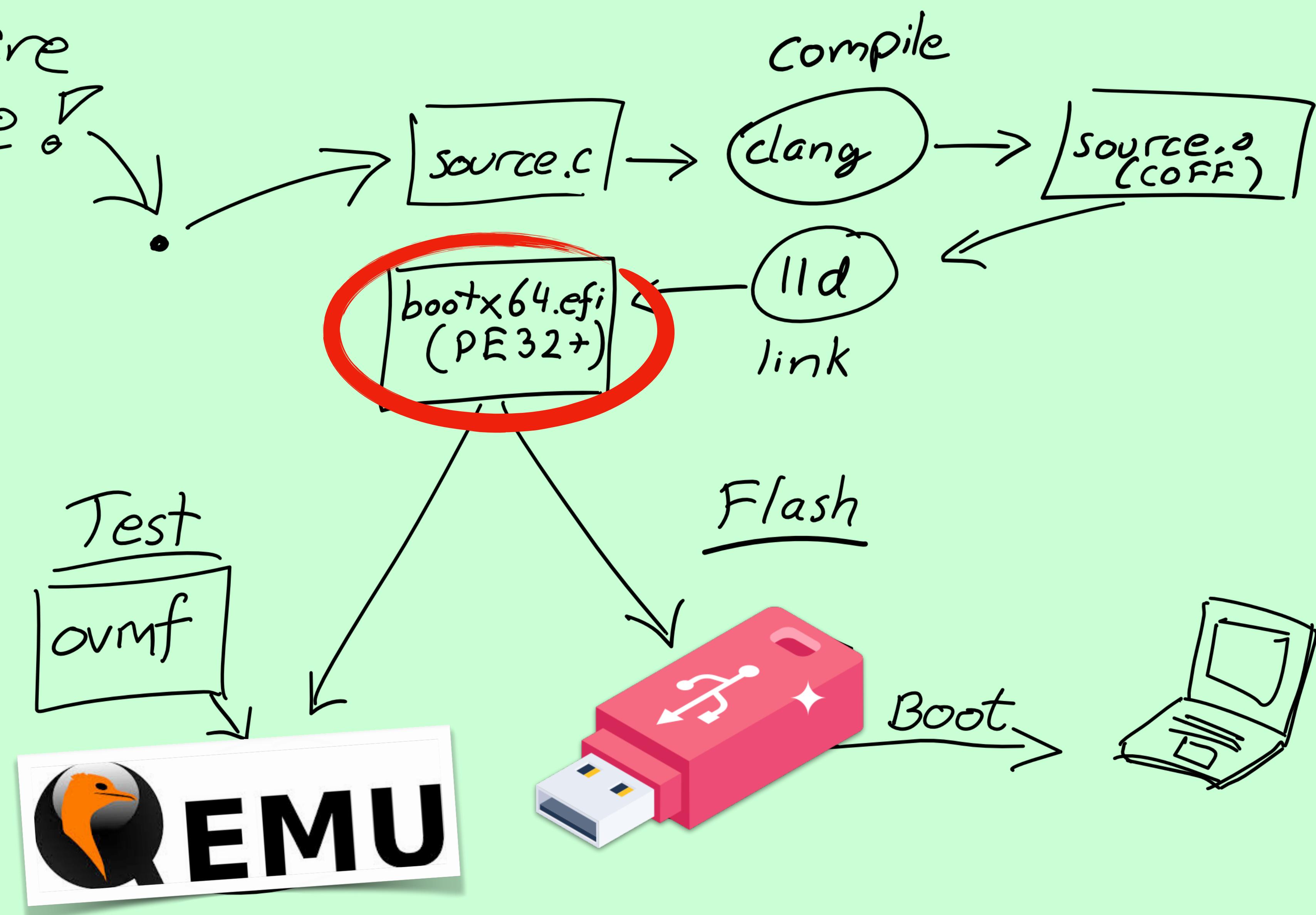
LLVM



The process



You are
here!



Anatomy of a UEFI-application

- Main function
 - `*system_table` - entry point to everything UEFI
 - Console input/output
 - Boot services
 - Query/address protocols, Memory allocation, Event handling, ...
 - Runtime services
 - Query runtime information from hardware devices, e.g. system time

```
examples > hello > C anatomy.c > ...
1 // Example anatomy of a UEFI-application
2 #include "lil_uefi/lil_uefi.h"
3 #include <stddef.h> // for NULL
4
5 // Entry point:
6 // https://uefi.org/specs/UEFI/2.10/04_EFI_System_Table.html#efi-image-entry-
7 EFI_UINTN EfiMain(EFI_HANDLE handle, EFI_SYSTEM_TABLE *system_table)
8 {
9     EFI_STATUS status = 0;
10
11     // Prevent automatic 5min abort
12     system_table->BootServices->SetWatchdogTimer(0, 0, 0, NULL);
13
14     // Locate protocol(s) of choice -- e.g. graphics output device
15     EFI_GUID gfx_out_guid = EFI_GRAPHICS_OUTPUT_PROTOCOL_GUID;
16     struct EFI_GRAPHICS_OUTPUT_PROTOCOL *gfx_out_prot;
17     status = system_table->BootServices->LocateProtocol(
18         &gfx_out_guid,
19         0, // The first graphics device
20         (void **) &gfx_out_prot);
21
22     if (!status)
23     {
24         return status;
25     }
26
27     // Access the console output
28     EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL *out = system_table->ConOut;
29     out->ClearScreen(out);
30
31     out->SetCursorPosition(out, 0, 0);
32     out->OutputString(out, L"Hello, world!");
33
34     return status;
35 }
```

Yes. It's pointers and
structs all the way down

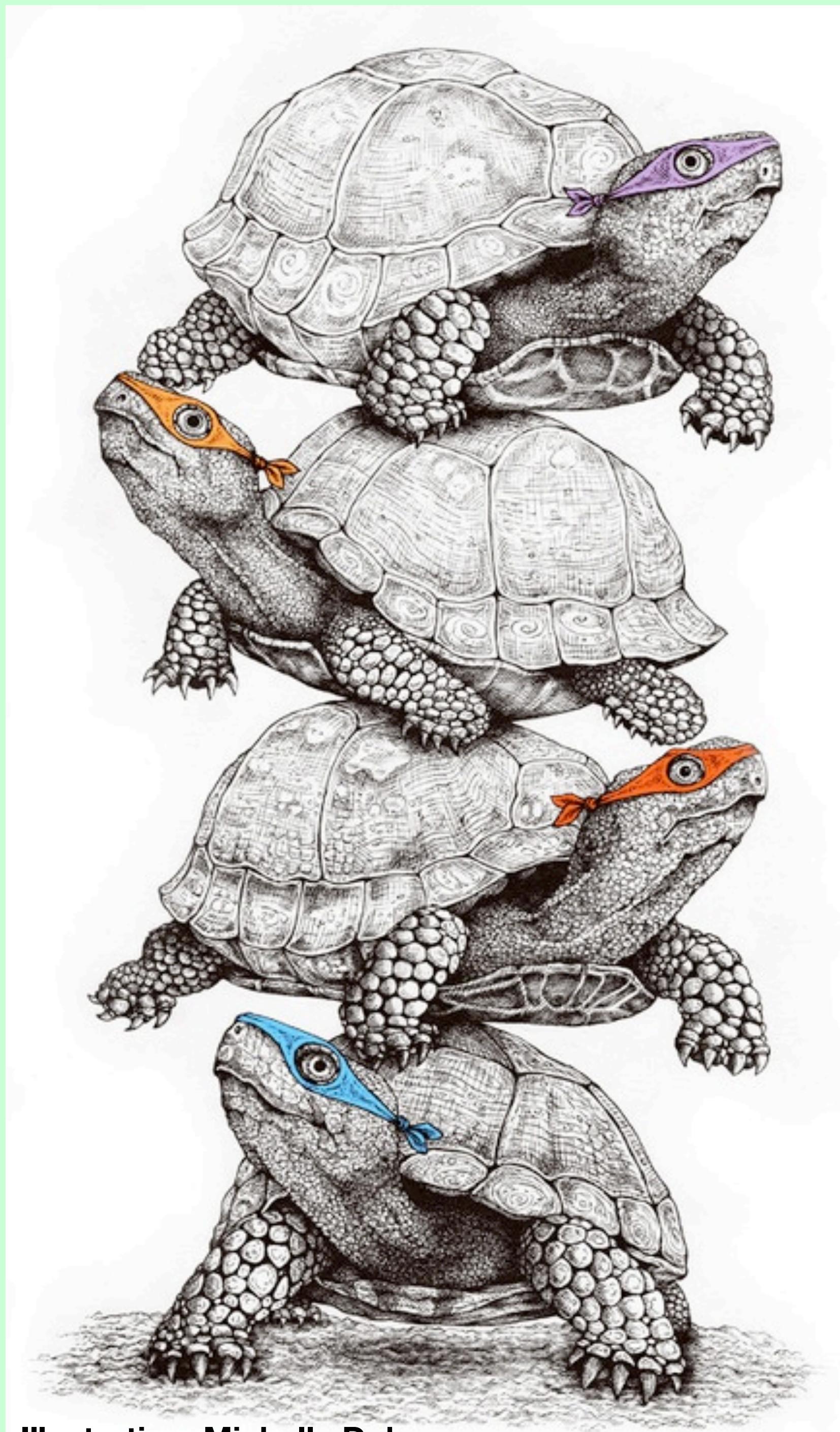


Illustration: Michelle Duke

It's demo-time!
Let's dive in...

Couple of tips / nice to knows

- PC-users: Disable secure boot
- There's no "standard library" by default
- For long-running apps: disable automatic reboot watchdog (see `hello.c`)
- Not all systems are born equal:
 - The exact behaviour and capabilities available might differ from system to system. E.g. `qemu` and some PCs will render console output to the screen, whereas MacBook Pros will not.
 - `qemu` w/ OVMF has no pointer device drivers loaded at boot time, whereas all PCs and MBPs I've tested do.

Further reading

- **UEFI specification**
<https://uefi.org/specifications>
- **lil_uefi by Allen Webster and Ryan Fleury**
<https://handmade.network/p/308/lil-uefi/>
- **osdev.org**
<http://wiki.osdev.org/UEFI>
- **TianoCore / EDK II - “EFI Development Kit” - Reference-impl by Intel**
<https://www.tianocore.org/> / <https://github.com/tianocore/edk2>
OVMF is borrowed from this
- **uefi-simple**
<https://github.com/pbatard/uefi-simple>

Questions?



Now go do
something
good fun!

