HOWTO: Boot an OS

By Camille Lecuyer

LSE Week - July 17 2013

PRESENTATION

- EPITA 2014 - GISTRE
- Not LSE team

EPITA

ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

SUMMARY

- BIOS
- UEFI
- Boot a Linux kernel
- Boot a Multiboot compliant kernel

BIOS

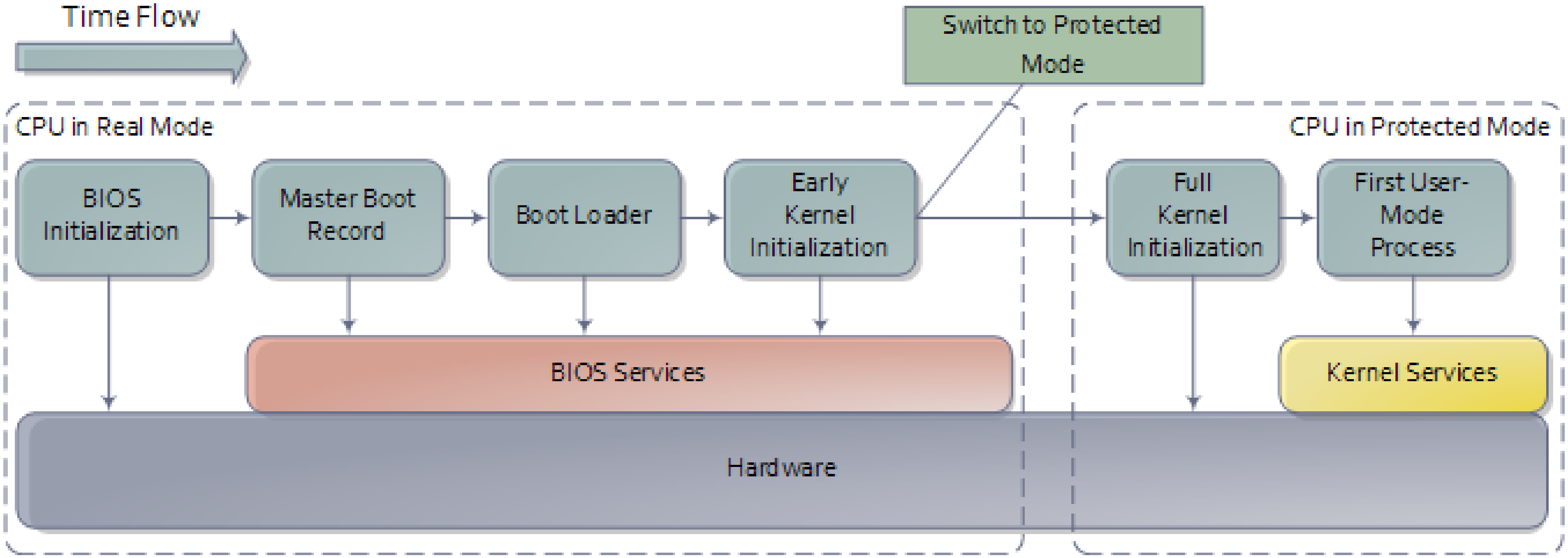
OVERVIEW

- Basic Input Output System
- First used in the CP/M operating system in 1975 => very old!
- Widely used in compatible IBM PC (since 1981)
- Still present today in computers but dying
- Replaced by UEFI

TYPICAL COMPUTER BOOT

- CPU load 0xFFFF0 (reset vector)
- POST (power on self test)

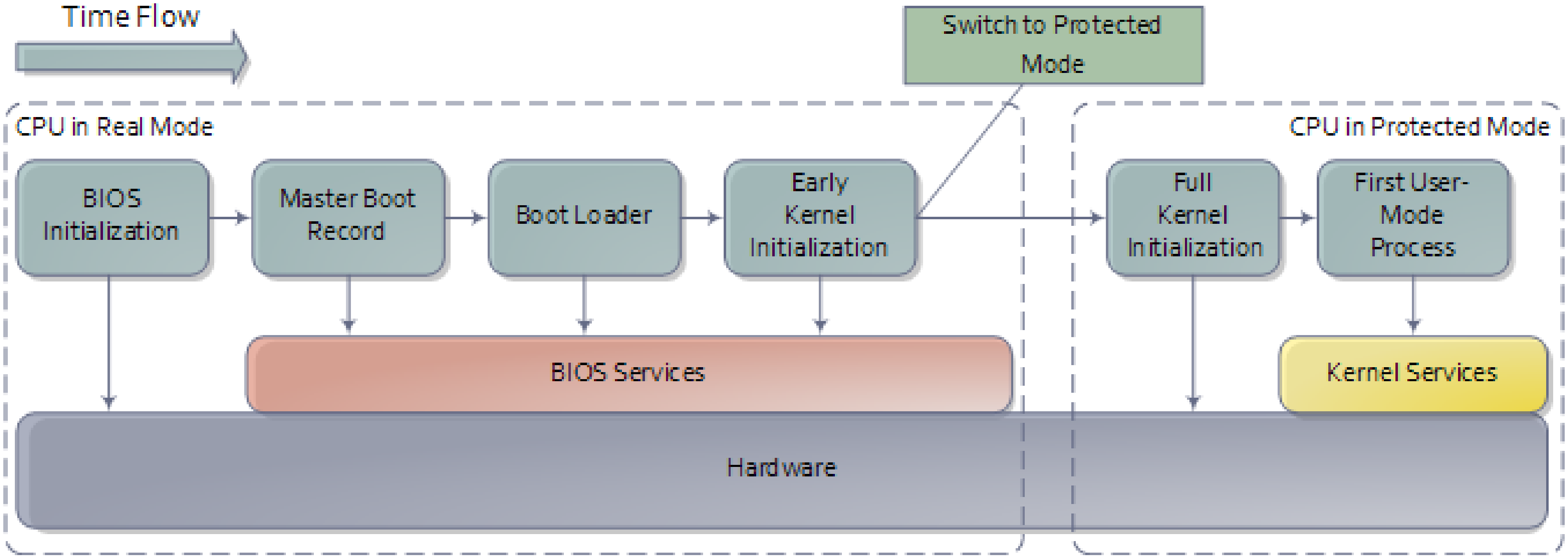
TYPICAL COMPUTER BOOT



TYPICAL COMPUTER BOOT

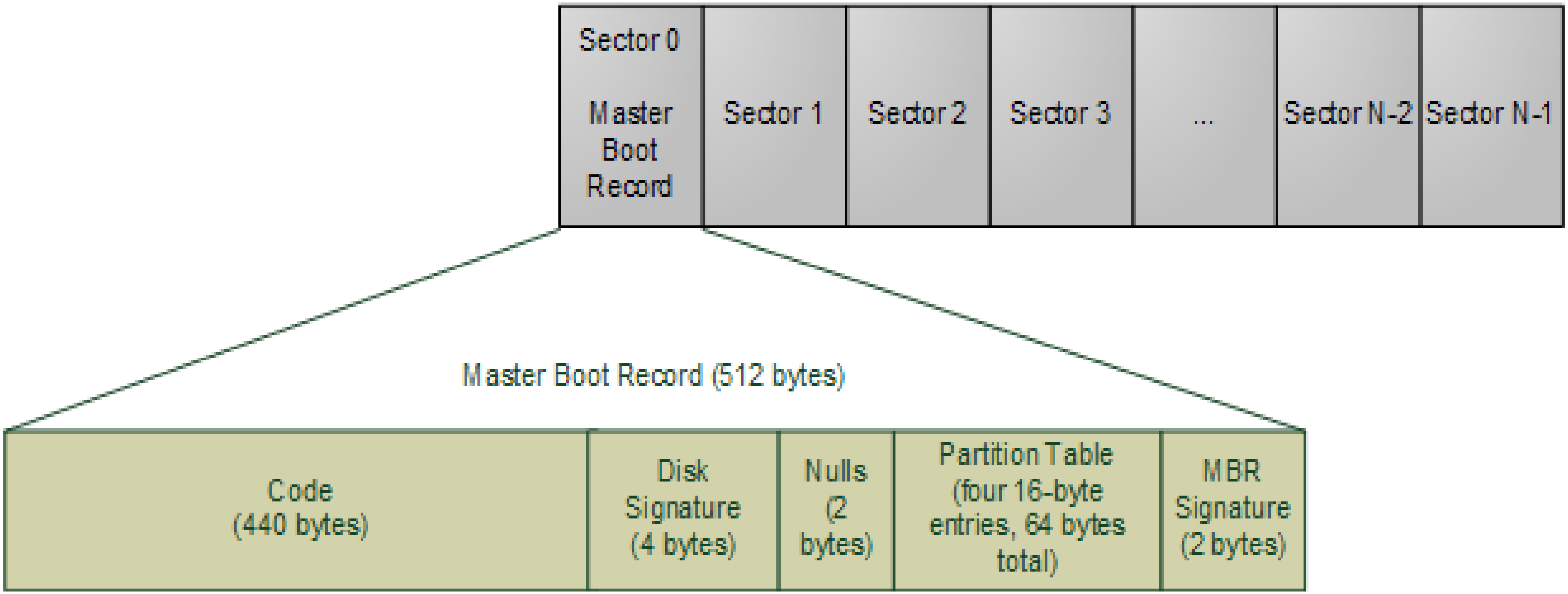
- Try to find a bootable device:
- Select a device
  - Load its first sector (MBR) at 0x7C00
  - Check signature: 0x55 0xAA
  - If found, jump at 0x7C00

TYPICAL COMPUTER BOOT



MBR (MASTER BOOT RECORD)

N-sector disk drive. Each sector has 512 bytes.



Sector 0 Master Boot Record | Sector 1 | Sector 2 | Sector 3 | ... | Sector N-2 | Sector N-1

Master Boot Record (512 bytes)

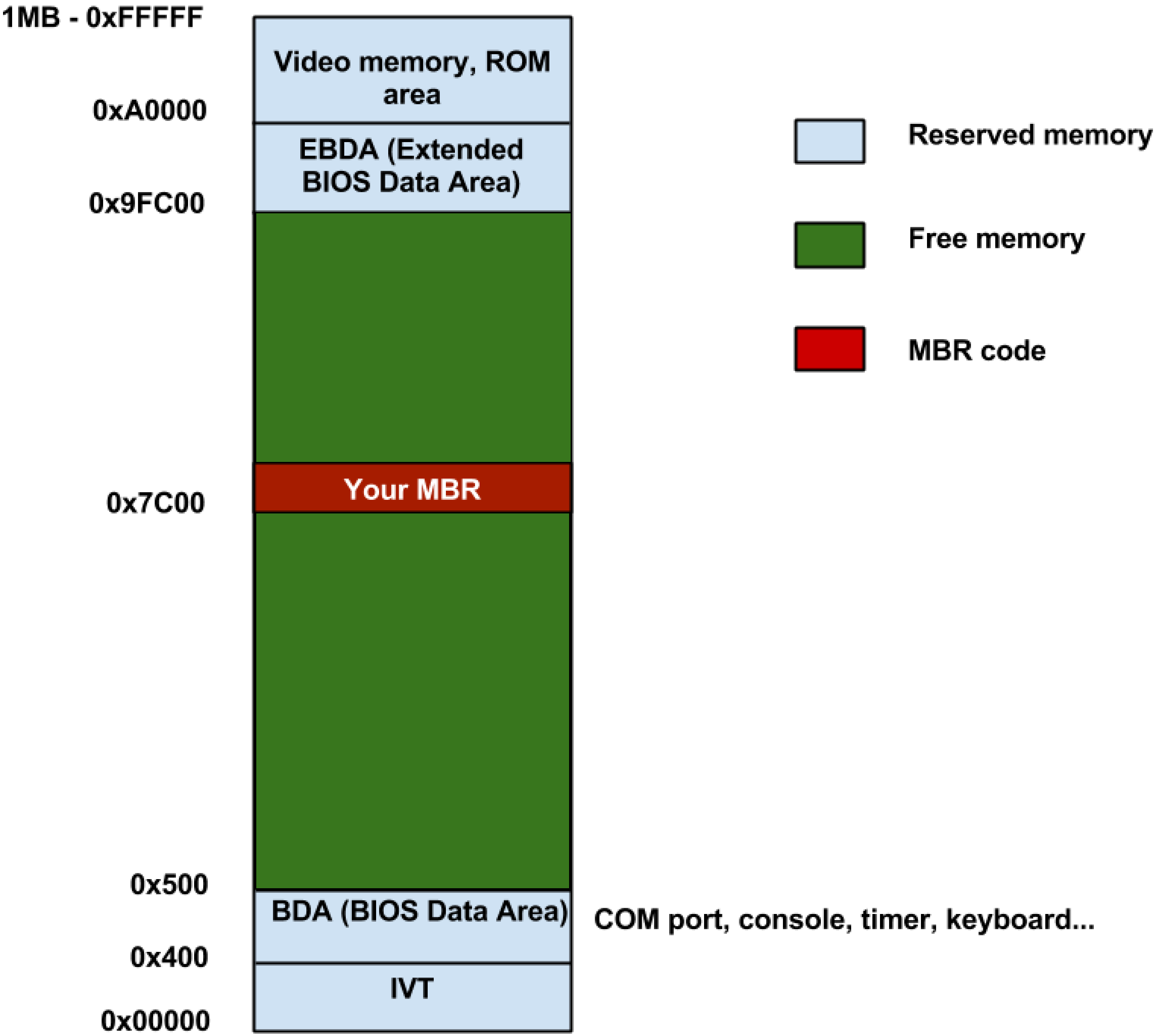
Code (440 bytes) | Disk Signature (4 bytes) | Nulls (2 bytes) | Partition Table (four 16-byte entries, 64 bytes total) | MBR Signature (2 bytes)

QUICK DEVELOPER VIEW

- First layer before the hardware
- Provides software interface for programmer
- Only 16 bit code (intel real mode)
- Only 1MB of memory reachable!
- ASM code
- Easy device access thanks to BIOS services
  - Display
  - Keyboad
  - Disks (LBA – Logical Block Access)
  - Memory mapping…
- Use interrupt system (ex: int $0x15)

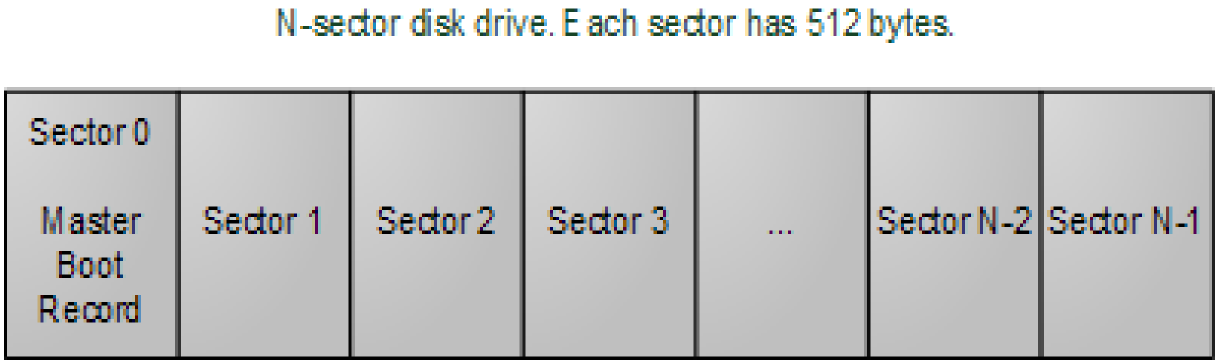
ENVIRONMENT ALMOST EMPTY

- Flat binary => no binary format like ELF
- No lib provided (only bios services)
- Things to setup:
  - Stack
  - Initialize registers
- Memory mapping (keep it clear in mind)


1MB - 0xFFFFF
Video memory, ROM area
0xA0000
EBDA (Extended BIOS Data Area)
0x9FC00
Your MBR
0x7C00
0x500
BDA (BIOS Data Area)
COM port, console, timer, keyboard...
0x400
IVT
0x00000
🟦 Reserved memory
🟩 Free memory
🟥 MBR code

TYPICAL BOOTLOADER DESIGN

- Stage1
- Stage2
- Grub: stage 1.5
- Switch between real mode and protected mode



N-sector disk drive. Each sector has 512 bytes.

Sector 0 Master Boot Record	Sector 1	Sector 2	Sector 3	...	Sector N-2	Sector N-1

UEFI

Unified Extensible Firmware Interface

HISTORY

- 2001: EFI Spec started for Intel Itanium
- 2005: Stop of development at v1.10 but Unified EFI Forum continue the project as UEFI.
  - Intel, AMD, AMI, Apple, Dell, HP, IBM, Microsoft, Phoenix...
- 2007: v2.1
- 2009: Add ARM processor binding to UEFI
- 2013: v2.4

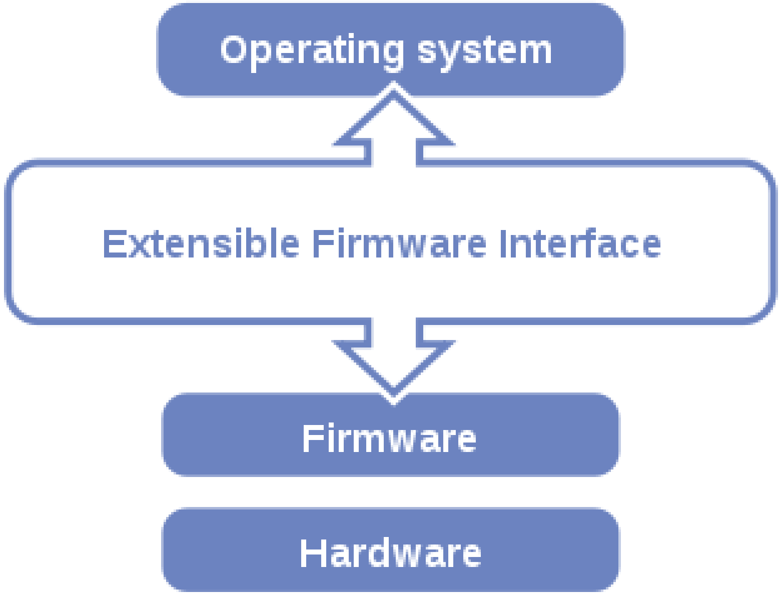
- http://www.uefi.org/specs/

WHY UEFI?

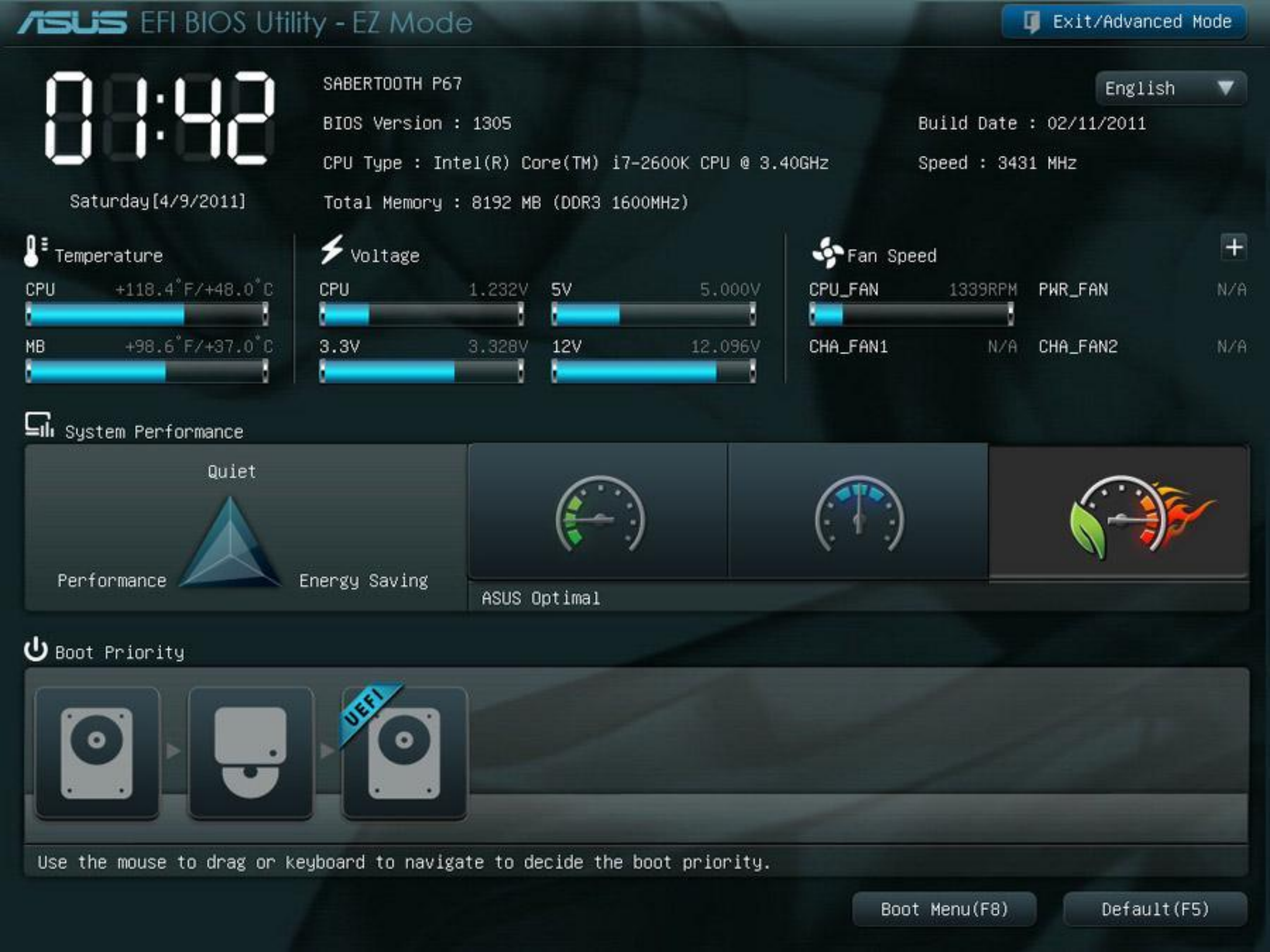
- Replace the old BIOS
- Load 32 or 64 bit code from the start (and not 16 bit => all memory available!)
- C programming
- Provides a wide framwork
- Load PE32+ programs
- All the environment is ready
- GPT
- Secure Boot: signed binary by trusted user
- TCP/IP

UEFI GOAL

- "The purpose of the UEFI interfaces is to define a common boot environment abstraction for use by loaded UEFI images, which include UEFI drivers, UEFI applications, and UEFI OS loaders."
  - UEFI Spec



USER VIEW…



ASUS EFI BIOS Utility - EZ Mode

01:42

Saturday[4/9/2011]

SABERTOOTH P67

BIOS Version : 1305

Build Date : 02/11/2011

CPU Type : Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz

Speed : 3431 MHz

Total Memory : 8192 MB (DDR3 1600MHz)

Temperature

CPU +118.4˚F/+48.0˚C

MB +98.6˚F/+37.0˚C

Voltage

CPU 1.232V

5V 5.000V

3.3V 3.328V

12V 12.096V

Fan Speed

CPU_FAN 1339RPM

PWR_FAN N/A

CHA_FAN1 N/A

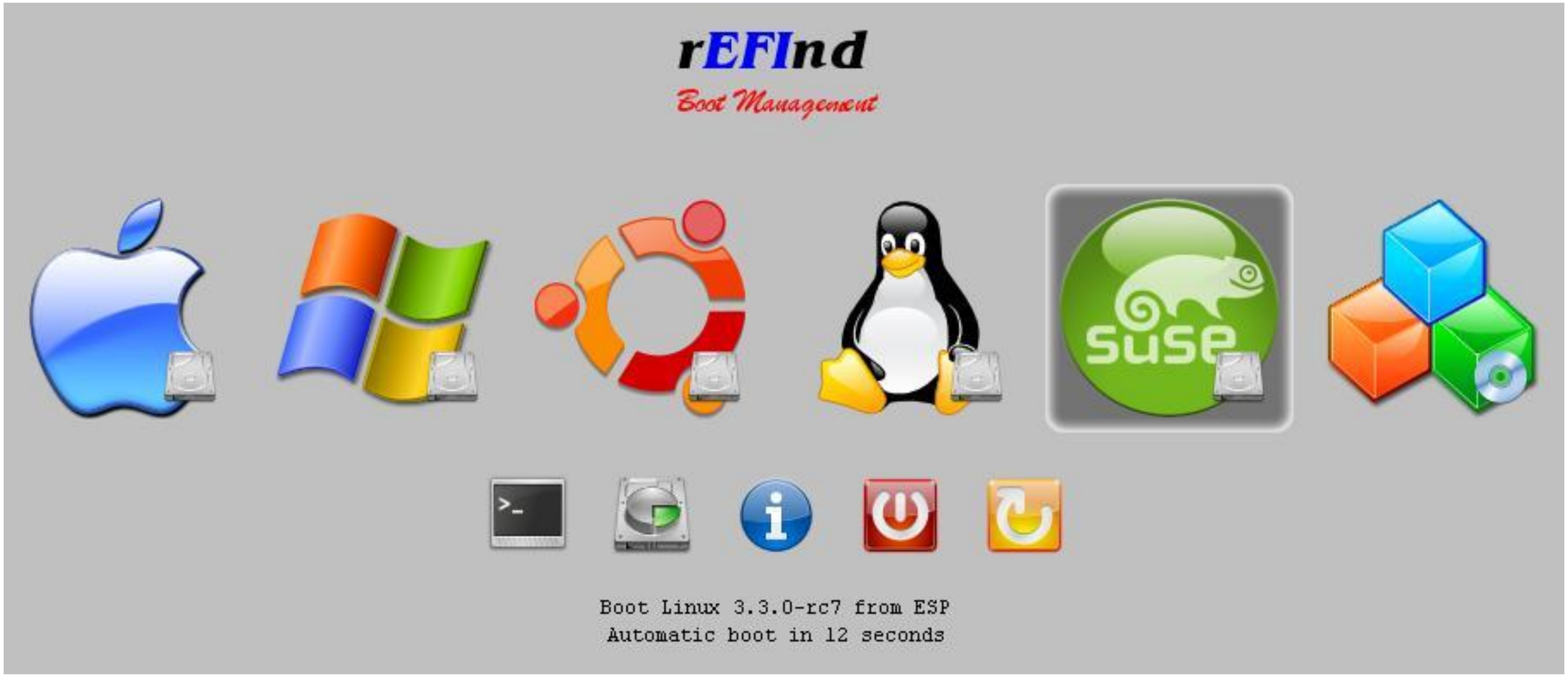
CHA_FAN2 N/A

System Performance

ASUS Optimal

Boot Priority

Use the mouse to drag or keyboard to navigate to decide the boot priority.



Boot Linux 3.3.0-rc7 from ESP
Automatic boot in 12 seconds

UEFI SPREAD THE WORLD

- Present in almost all new computers
- Present in Apple's Mac

OS SUPPORT

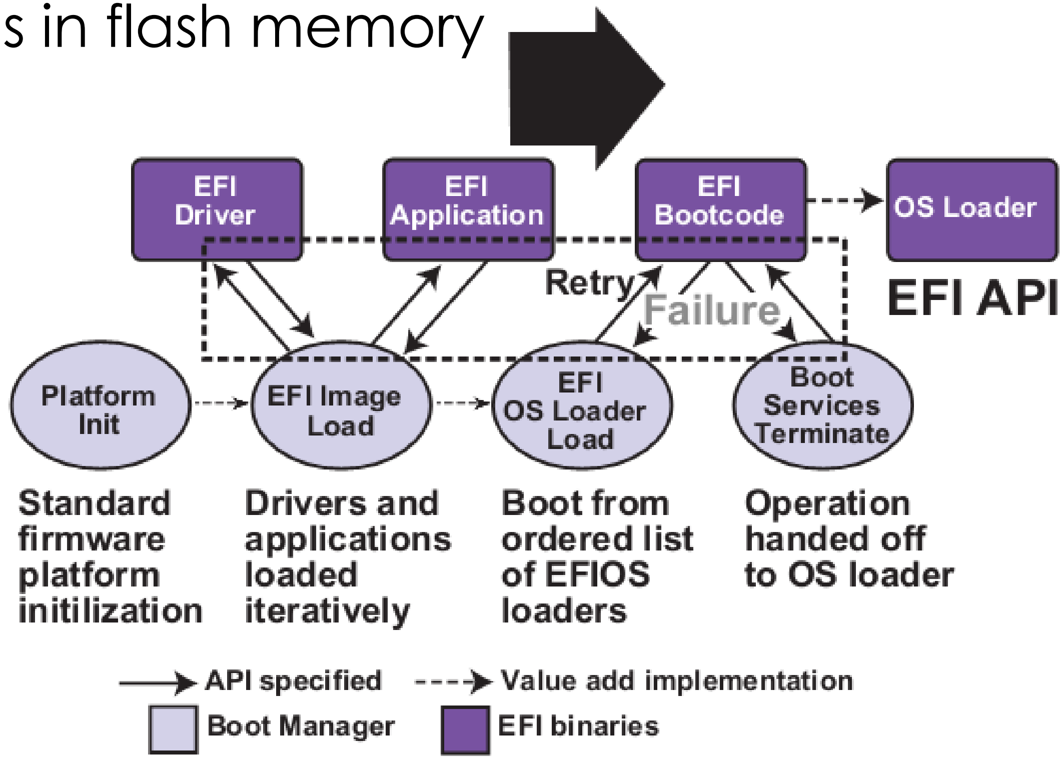
- Mac OS X: EFI 1.10, but only 32bit
- Windows: since Vista SP1 on 64bit versions (
- Linux:
  - With a bootloader supporting uefi
    - Refind, Gummiboot, or GRUB, elilo
  - With **EFI STUB**

NVRAM

- Internal memory used to store variables
- Contain file to boot and boot order
- Avaliable under linux in /sys/firmware/efi/vars/ thanks to efivar sysfs linux module
    - Defined in linuxrepo/drivers/firmware/efi/efivars.c

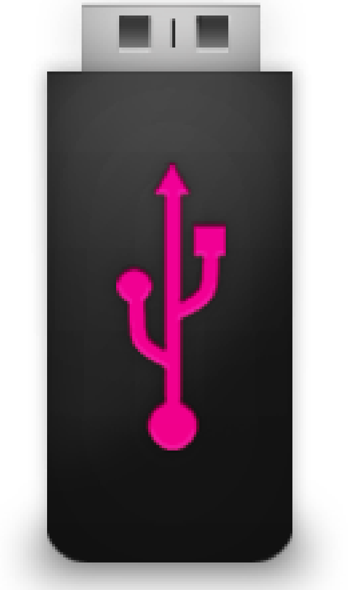
UEFI BOOT PROCESS

- Can read partition tables and filesystems
- Load EFI/boot/bootx64.efi or boot loader whose filename is in flash memory



HOW TO CREATE A BOOTABLE DISK?

- Fat32 partition
- Add your bootloader into
  - /EFI/boot/bootx64.efi
- Plug
- It works!
- No need of a MBR



UEFI PROGRAM

- PE32+ file with modified SubSystem field (10, 11, 12)
- UEFI Application
  - Simple application (shell, file editor, memtest, change efi variables...)
  - OS loader
- UEFI Boot service driver
- UEFI runtime driver

BOOT SERVICIES VS RUNTIME SERVICES

- Boot services:
  - Event, timer
  - Memory allocation
  - Driver handle
  - Image services (load, start, exit…)
  - **ExitBootServices**(): think to GetMemoryMap()
  - Functions available before ExitBootServices() is called
- Runtime services:
  - Variable
  - Time
  - Reset

EBC – EFI BYTE CODE VIRTUAL MACHINE

- Provides platform and processor independent boot mecanism
- Facilitate the removal of legacy infrastructure

TIANOCORE

- Provides SDK for UEFI
- Open source implementation of a UEFI firmware
  - Works with Qemu

HOW TO CODE?

- Under Windows: Use Tiano project with Visual studio
- Under Linux: Use GNU-efi
  - UEFI and Linux ABI doesn't match:
  - We use wrappers
- Get the spec!

GNU-EFI

- Provide headers and wrappers
- Provide additional library
- Use objcopy's efi feature
    - objcopy --efi-app-x86_64
- .o → .so → .efi

HOW TO CODE?

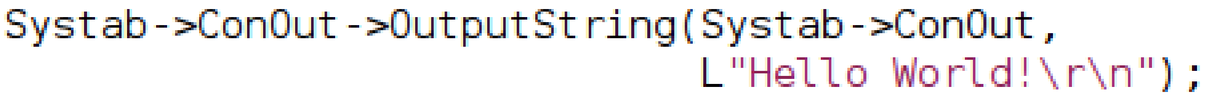
```
EFI_STATUS efi_main (EFI_HANDLE image,
                     EFI_SYSTEM_TABLE *systab)
```

- All you need is in the system table:
  - Console access
  - Boot services
  - Runtime services

- Functions pointer

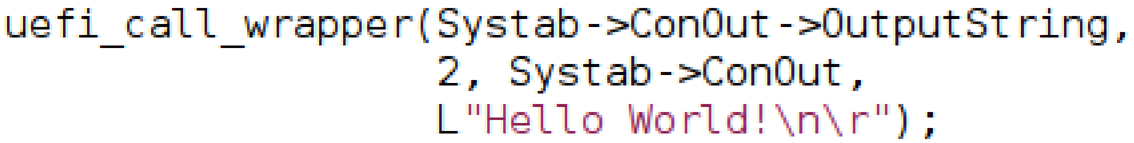
SIMPLE HELLO WORLD

- With TianoCore SDK:



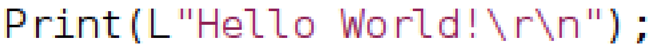
```
Systab->ConOut->OutputString(Systab->ConOut,
                             L"Hello World!\r\n");
```

- With GNU-EFI:



```
uefi_call_wrapper(Systab->ConOut->OutputString,
                  2, Systab->ConOut,
                  L"Hello World!\n\r");
```

- With efilib:



```
Print(L"Hello World!\r\n");
```

LOAD A LINUX KERNEL ON X86

LINUX KERNEL

- Originaly booted from a floppy disk with integrated bootloader
- Today, we have to use a bootloader
- We use an initramfs (aka initrd, module)
- Multiple entry point:
  - 16 bit code (real mode)
  - 32 bit
  - 64 bit
  - UEFI boot Stub

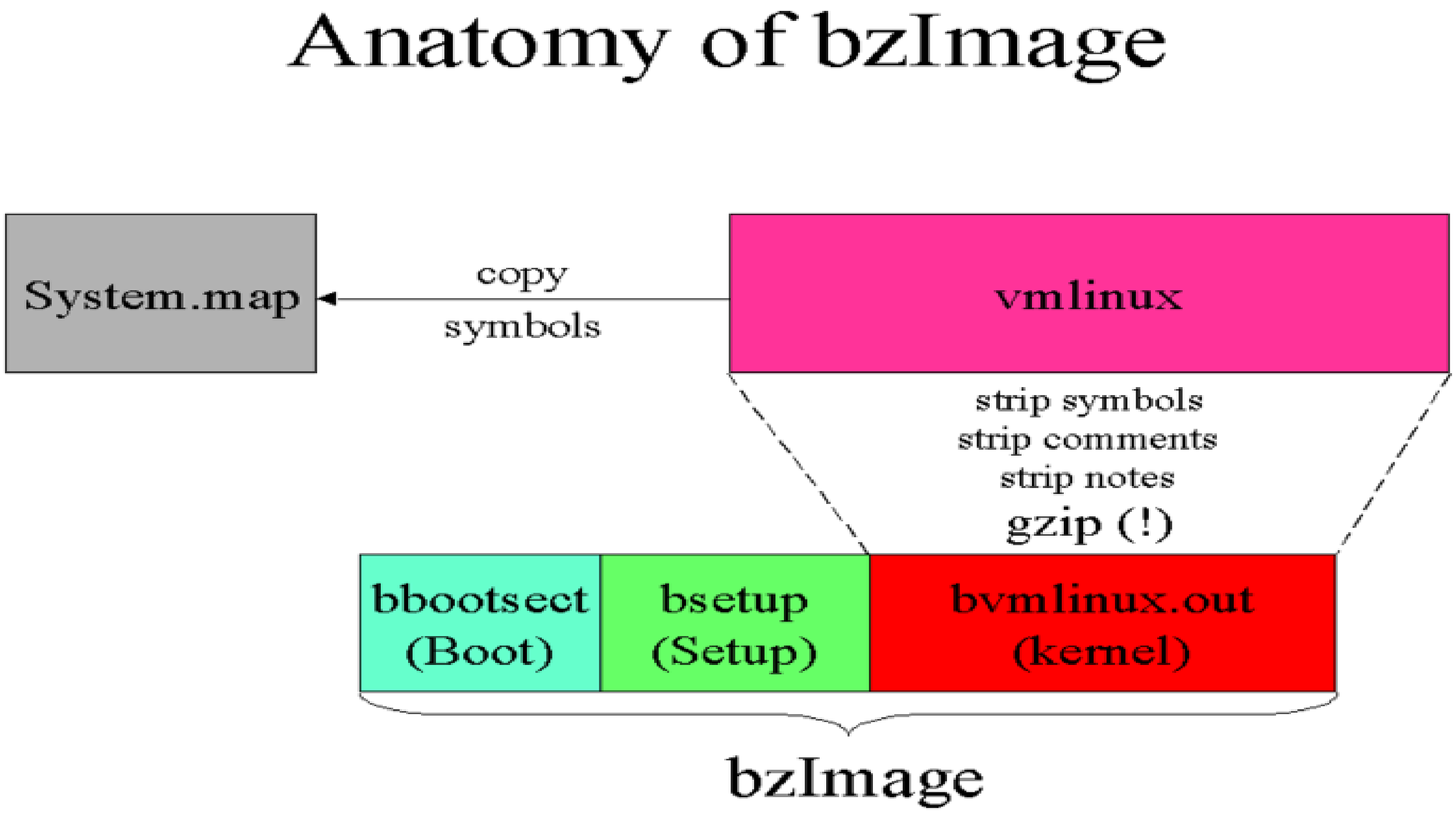
PROTOCOL HISTORY

- Boot protocols evolve across linux versions:

- < 2.0 (linux 1.3.73): only Image and zImage
- 2.0: bzImage and initrd
- 2.11 (linux 3.6): add fields for EFI
- 2.12 (linux 3.8): allow to load a kernel over 4GB in 64bit mode.

- Cf linuxrepo/Documentation/x86/boot.txt

KERNEL IMAGE FORMAT



- Also exist Image and zImage
- Cf linux/arch/x86/boot/tools/build.c

REAL MODE KERNEL HEADER

- Structure given to linux (struct setup_header)
- Filled by the bootloader
- Legacy structure
    - sector magic number
    - Protocol version
    - Kernel version
    - Initramfs info
    - Kernel command line
    - Hooks
- Description under Documentation/x86/boot.txt
- arch/x86/include/uapi/asm/bootparam.h

REAL MODE CODE

- 16 bit code – asm and C
- Fill struct boot_params
- Init env (lot of bios call):
  - Early console and serial
  - Check cpu
  - Detect memory (e820)
  - Enable keyboad
- Go in protected mode (pm.c and pmjump.S)
- Entry point : linux/arch/x86/boot/header.S

PROTECTED MODE

- Set GDT, IDT, paging for next step
- Linux/arch/x86/kernel/head_{32,64}.S

EFI STUB

- Since linux 3.3
- Fill boot_params and setup_header structures with efi call
- efi_main
    - Setup graphics
    - Allocate memory for stucture (GDT, IDT…)
    - ExitBootServices
    - Setup GDT, IDT (empty for now)
    - Load initramfs from cmdline (initrd=/EFI/linux/initramfs.img) with efi boot services
- Jump on 64bit code

LOAD A MULTIBOOT COMPLIANT KERNEL ON X86

MULTIBOOT SPECIFICATION

- 1995
- Configure system at boot time
- Handle modules
- Structures and machine state

- Easy to use for your first kernel

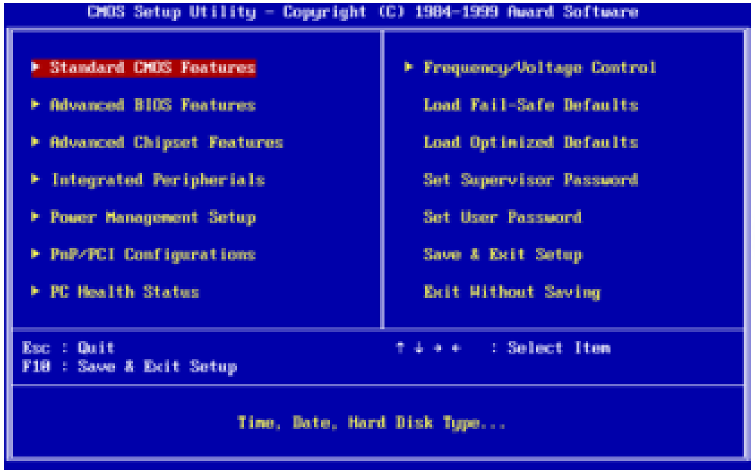
- http://www.gnu.org/software/
  grub/manual/multiboot/multiboot.html

MULTIBOOT STRUCTURES

- Multiboot header:
  - Magic number
  - Flags
- Multiboot info:
  - Memory mapping
  - Cmdline
  - Module info

CONCLUSION

- Dev feedback
- BIOS VS UEFI



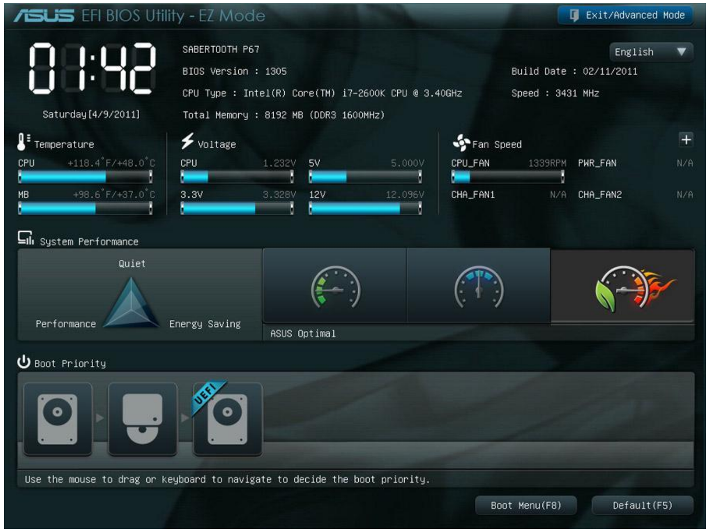
```
CMOS Setup Utility - Copyright (C) 1984-1999 Award Software

▸ Standard CMOS Features          ▸ Frequency/Voltage Control
▸ Advanced BIOS Features            Load Fail-Safe Defaults
▸ Advanced Chipset Features         Load Optimized Defaults
▸ Integrated Peripherials           Set Supervisor Password
▸ Power Management Setup            Set User Password
▸ PnP/PCI Configurations            Save & Exit Setup
▸ PC Health Status                  Exit Without Saving

Esc : Quit                        ↑ ↓ → ←   : Select Item
F10 : Save & Exit Setup

                    Time, Date, Hard Disk Type...
```

VS



```
01:42
Saturday[4/9/2011]

SABERTOOTH P67
BIOS Version : 1305                                          Build Date : 02/11/2011
CPU Type : Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz          Speed : 3431 MHz
Total Memory : 8192 MB (DDR3 1600MHz)

Temperature                  Voltage                               Fan Speed
CPU   +118.4°F/+48.0°C       CPU   1.232V     5V    5.000V         CPU_FAN  1339RPM   PWR_FAN   N/A
MB    +98.6°F/+37.0°C        3.3V  3.328V     12V   12.096V        CHA_FAN1 N/A       CHA_FAN2  N/A

System Performance
        Quiet
Performance   Energy Saving         ASUS Optimal

Boot Priority
Use the mouse to drag or keyboard to navigate to decide the boot priority.

Boot Menu(F8)     Default(F5)
```

CONTACT AND LINKS

- camille.lecuyer@gmail.com

- git@bitbucket.org:cakou/cb.git

- Bootloader from scratch: http://www.cs.cmu.edu/~410-s07/p4/p4-boot.pdf

- http://www.mcamafia.de/pdf/pdfref.htm
- http://www.phoenix.com/resources/specs-bbs101.pdf

- http://x86asm.net/articles/uefi-programming-first-steps/index.html
- http://www.rodsbooks.com/efi-bootloaders/

Questions?