Linux Boot Process

Nassim Eddequiouaq

LSE Summer Week 2015

Why does boot matter ?

No boot... No boot !

OS uses evolving hardware features

Faster and more secure please

LSE Security System
Laboratory of Epita

What does Linux need ?

Hardware initialization

Bootloader loading the kernel

Initialize arch-dependent and basic features

Setup interrupts, memory management, SMP...

Boot logic

Before the operating system

Backward compatibility

OS boot process is specific

Setup and hooking features

... Linux Boot Protocol

Boot steps

The bootloader

Kernel setup

Long mode

Decompression and jump to kernel code

Bootloaders

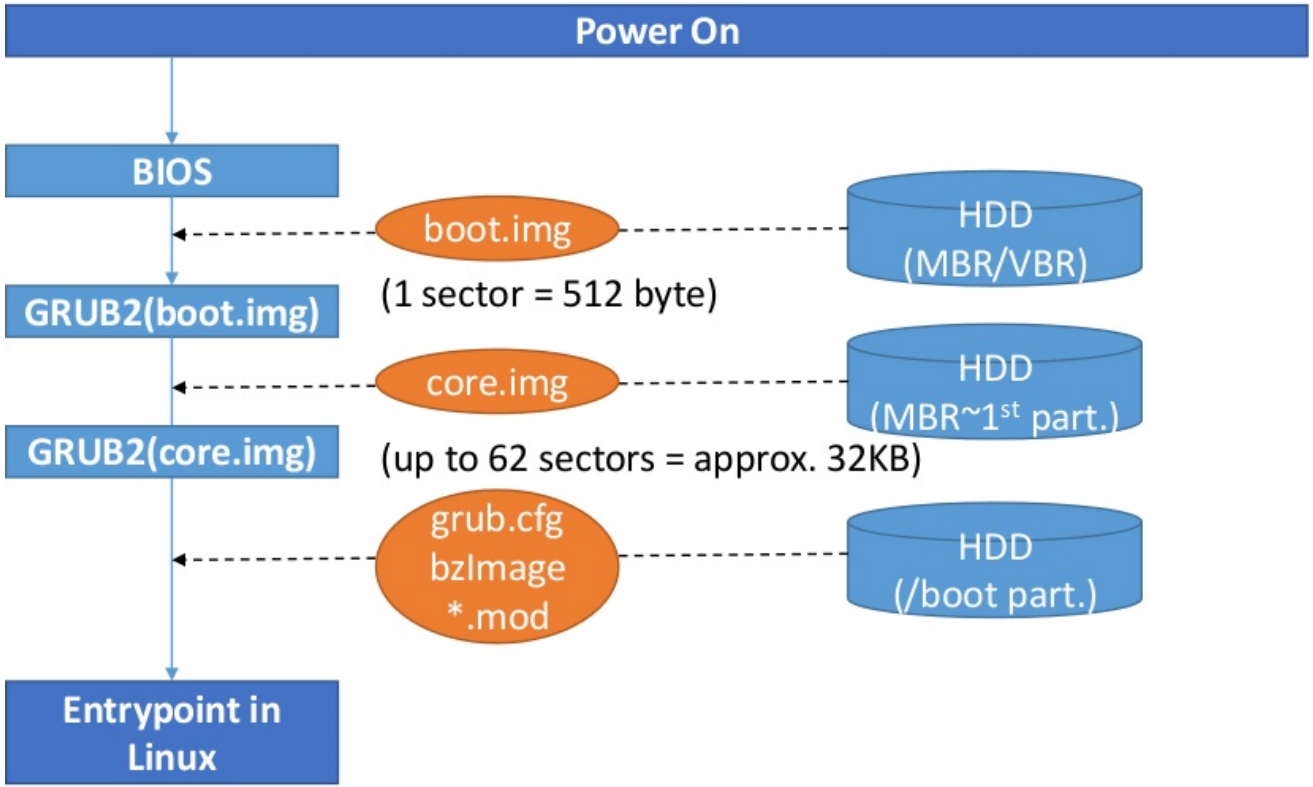
Responsible for loading and transferring control to the kernel

- UEFI
- Legacy

- LILO
- GRUB2
- SYSLINUX

LSE Security System
Laboratory of Epita

GRUB2 boot sequence



LSE Security System Laboratory of Epita

x86 Architecture

4 Modes :

- Real mode
- Protected mode
- V8086 mode
- Long mode

Linux Boot Protocol

Build-time parameters :

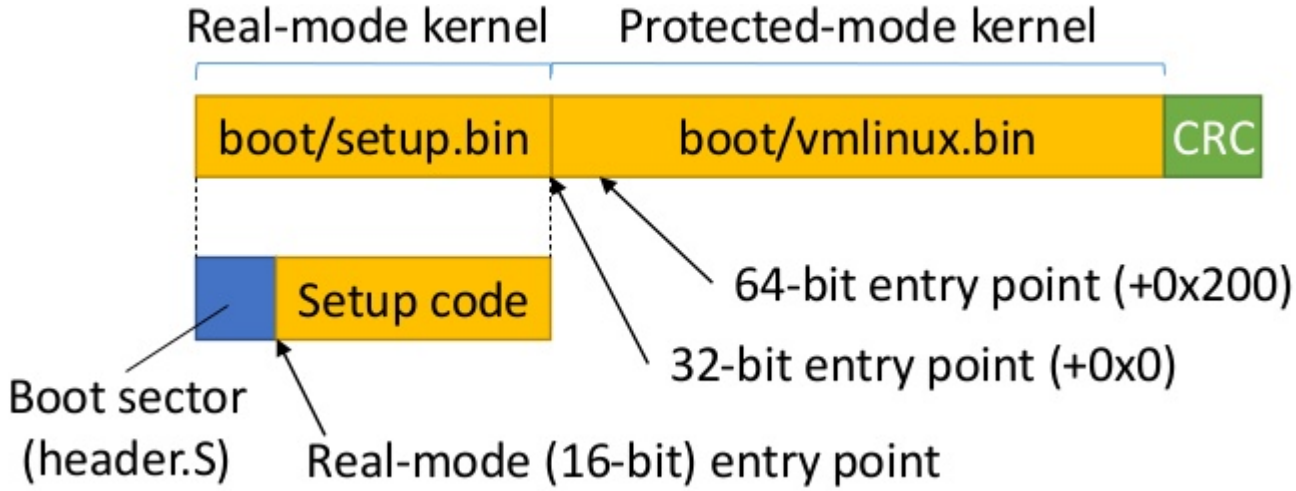
- setup code size

Bootloading-time parameters :

- command-line parameters and size
- initrd max address

Linux image format

The bzImage



Linux boot protocol for bzImage

4 entry points :

1. Real mode (16-bit)
2. Protected mode (32-bit)
3. Long mode (64-bit)
4. Final kernel entry (vmlinux decompressed)

LSE Security System
Laboratory of Epita

The bootloader

Implement the Linux Boot Protocol

Get and load kernel modules

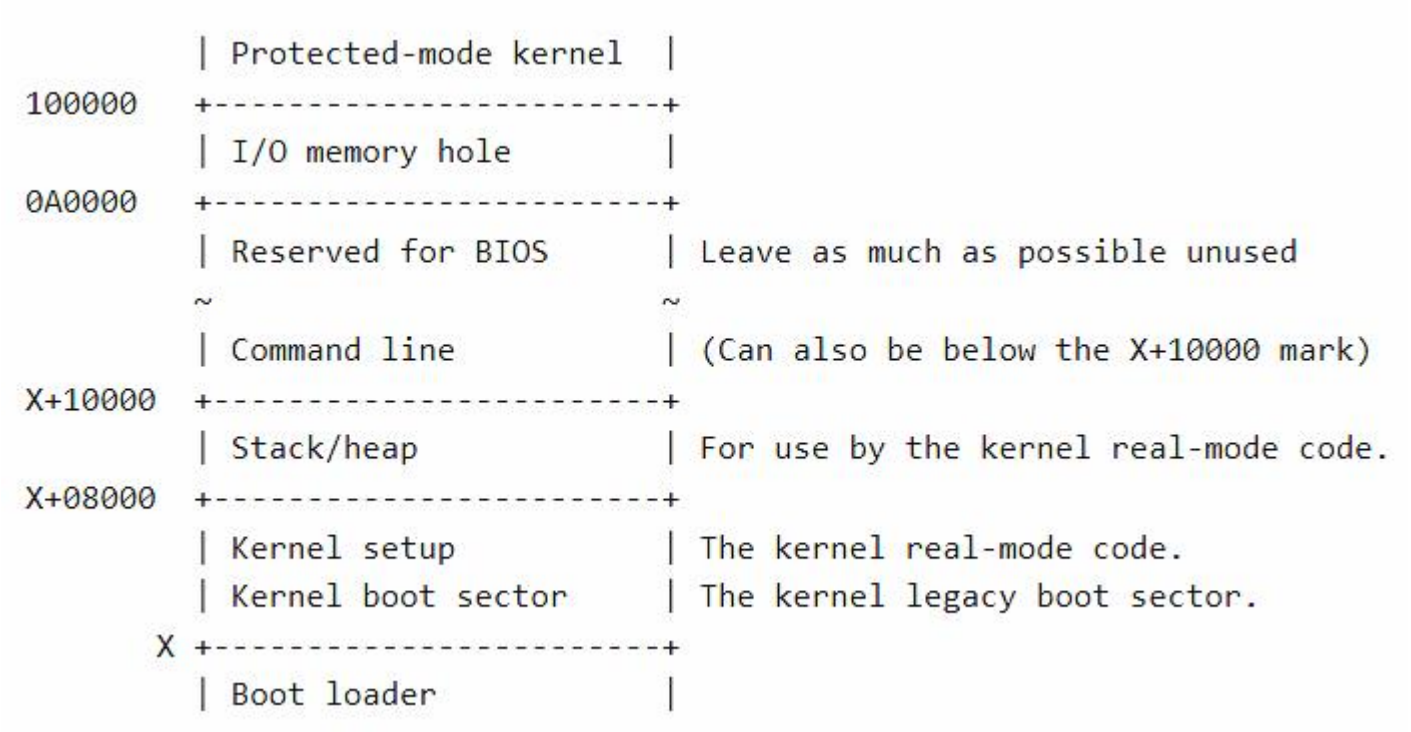
Fill the kernel setup header at the right address

Jump to the kernel entry point

Kernel setup header

Offset /Size	Proto	Name	Meaning
01F1/1	ALL(1	setup_sects	The size of the setup in sectors
01F2/2	ALL	root_flags	If set, the root is mounted readonly
01F4/4	2.04+(2	syssize	The size of the 32-bit code in 16-byte paras
01F8/2	ALL	ram_size	DO NOT USE - for bootsect.S use only
01FA/2	ALL	vid_mode	Video mode control
01FC/2	ALL	root_dev	Default root device number
01FE/2	ALL	boot_flag	0xAA55 magic number
0200/2	2.00+	jump	Jump instruction
0202/4	2.00+	header	Magic signature "HdrS"
0206/2	2.00+	version	Boot protocol version supported
0208/4	2.00+	realmode_swtch	Boot loader hook (see below)

Kernel memory map



LSE Security System
Laboratory of Epita

Protocol Requirements

- Kernel usually loaded at 1MB (any position if relocatable, fixed if not)
- cs (and loaded GDT) must be __BOOT_CS
- ds, es and ss must be __BOOT_DS
- esi has the struct boot_params address
- ebp, edi and ebx must be 0
- Interrupt disabled

Finally in Linux

ifdef CONFIG_EFI_STUB then linux is a PE

_start and start_of_setup

Before boot main (legacy)

- ➔ check the segment registers
- ➔ setup a stack if needed
- ➔ setup the bss
- ➔ jump to boot main

LSE Security System
Laboratory of Epita

Problem

```
43          .global bootsect_start
44  bootsect_start:
45  #ifdef CONFIG_EFI_STUB
46          # "MZ", MS-DOS header
47          .byte 0x4d
48          .byte 0x5a
49  #endif
50
51          # Normalize the start address
52          ljmp    $BOOTSEG, $start2
```

```
65  msg_loop:
66          lodsb
67          andb    %al, %al
68          jz      bs_die
69          movb    $0xe, %ah
70          movw    $7, %bx
71          int     $0x10
72          jmp     msg_loop
73
74  bs_die:
75          # Allow the user to press a key, then reboot
76          xorw    %ax, %ax
77          int     $0x16
78          int     $0x19
79
80          # int 0x19 should never return.  In case it does anyway,
81          # invoke the BIOS reset code...
```

```
93  bugger_off_msg:
94          .ascii  "Use a boot loader.\r\n"
95          .ascii  "\n"
96          .ascii  "Remove disk and press any key to reboot...\r\n"
97          .byte   0
```

LSE Security System
Laboratory of Epita

Kernel setup (legacy)

- Copy boot header in the “zeropage”
- Init console and heap
- Check CPU and memory
- Queries (MCA, IST..)
- Set video mode
- Set Protected mode

Set protected mode (legacy)

Hook before leaving Real mode

Enable A20 gate

Reset coprocessor

Mask interrupts

GDT/IDT setup

LSE Security System
Laboratory of Epita

Hooks

Used in hostile environment (DOS) by the bootloader

boot_params.hdr.realmode_swtch 16-bit Real mode far subroutine that disables NMI

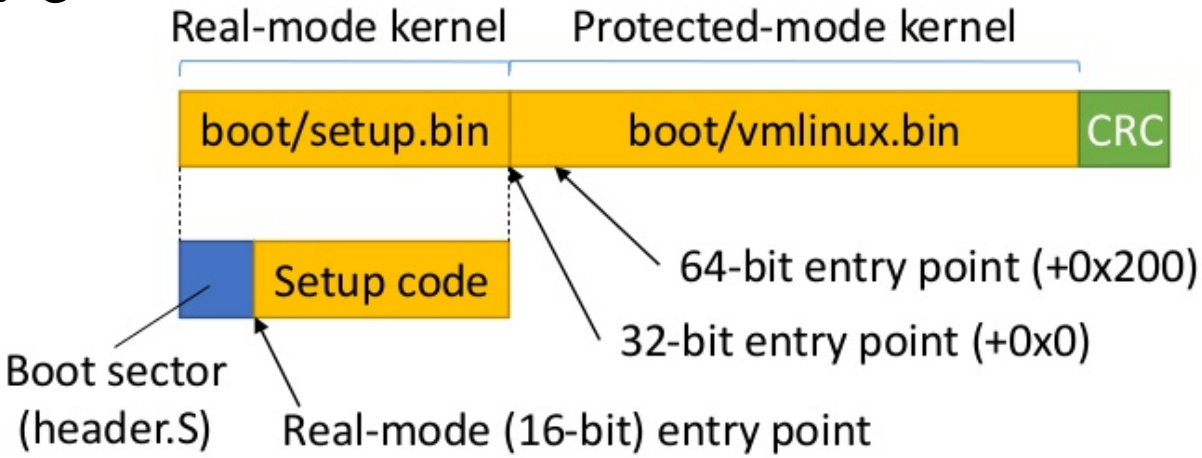
Last resort !

32-bit entry point

arch/x86/boot/compressed/head_64.S

startup_32 subroutine

Offset 0



32-bit to 64-bit

- Keep/Reload Segments
- Setup a stack and check long mode support
- Calculate reloc addr for decompression
- Update GDT for 64-bit and PAE in cr4

LSE Security System
Laboratory of Epita

Long mode

Native mode for x86_64 processors

Flat segmentation

Compatibility mode

64-bit registers, addresses and operands

MOAR REGISTERS

LSE Security System
Laboratory of Epita

Switch to 64-bit mode

Enable PAE (done)

Build PTs, update cr3 and EFER.LME in MSR

Enable paging

Jump to long mode

Set EFER.LME flag in MSR 0xC0000080 (done)

Push kernel cs

Push startup_64 routine

Set PG and PE in cr0

lret to startup_64

64-bit entry

startup_64 subroutine

Offset 0x200

Some bootloaders jump here directly (need identity mapped PT for kernel, zero page and commandline)

CONFIG_EFI_STUB

Allows bzImage to be loaded directly by EFI fw

Entry point is efi_pe_entry()

Setup boot params, startup_32

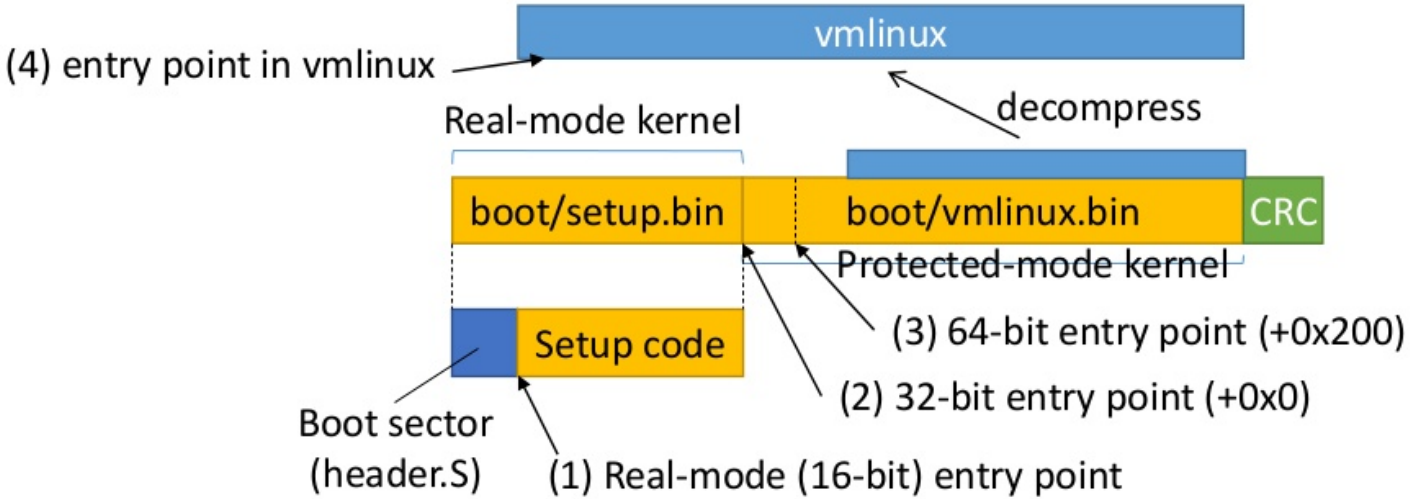
efi_main()

Kernel code decompression

- Compute decompressed kernel start address
- Setup a heap/stack and copy the compressed kernel
- Clear bss
- decompress_kernel()
- Parse ELF header, load sections and jump !

Yet another entry point

Real kernel code



LSE Security System
Laboratory of Epita

Beyond vmlinux entry point

Jump to startup_64

Fix physical addresses in the PT

Setup identity mapping

Enable PAE and PGE in cr4

Update cr3

start_kernel()

start_kernel()

Almost 150 initialization functions

Takes no args

Expect a specific and complex state

LSE Security System
Laboratory of Epita

What else before init ?

Locks

Threads

SMP

Scheduling

RCU

Proper interrupt handling, etc..

LSE Security System
Laboratory of Epita

EFI Runtime Services

efi_enabled(EFI_RUNTIME_SERVICES)

EFI Runtime Services Table

GetTime	Returns the current time and date, and the time-keeping capabilities of the platform.
SetTime	Sets the current local time and date information.
GetWakeupTime	Returns the current wakeup alarm clock setting.
SetWakeupTime	Sets the system wakeup alarm clock time.
SetVirtualAddressMap	Used by an OS loader to convert from physical addressing to virtual addressing.
ConvertPointer	Used by EFI components to convert internal pointers when switching to virtual addressing.
GetVariable	Returns the value of a variable.

LSE Security System
Laboratory of Epita

Conclusion

Not much recent changes to boot

UEFI bootloaders (GRUB2, Linux kernel EFI Stub)

Making an adaptive bootloader is not that easy

EFI offers a “good” and complete API

Question ?

(Thanks for listening !)

LSE Security System
Laboratory of Epita

References

- Linux 3.18
- Thanks @free-electrons.com for lxr
- GRUB 2
- Intel® 64 and IA-32 Architectures Software Developer Manual
- Linux Boot Protocol @ kernel.org
- UEFI wiki @ phoenix.com

LSE Security System
Laboratory of Epita