The Early System Start-Up Process

Group Presentation by: Tianyuan Liu, Caiwei He, Krishna Parasuram Srinivasan, Wenbin Xu
Boot Process

• Booting is the initialization of a computerized system

• In Linux, boot process is the multi-stage initialization process performed during starting a Linux installation
Boot Process

1. Firmware initialization

1. Execution of a boot loader

1. Loading and startup of a Linux kernel image

1. Execution of various startup scripts and daemons
Boot Process

- BIOS
  - Basic Input/Output System
  - Executes MBR

- MBR
  - Master Boot Record
  - Executes GRUB

- GRUB
  - Grand Unified Bootloader
  - Executes Kernel

- Kernel
  - Executes /sbin/init

- Init
  - Executes runlevel programs

- Runlevel
  - Runlevel programs are executed from /etc/rc.d/rc*.d
Overview - BIOS

- Stands for Basic Input/Output System
- Runs Power-On Self Tests (POST) and other integrity checks
- Searches, loads, and executes the boot loader program
Overview - MBR

- Stands for Master Root Record
- Loads and executes the GRUB boot loader
Overview - GRUB

- Stands for Grand Unified Bootloader
- If multiple kernel images are installed on your system, you can choose which one to be executed
- Loads and executes Kernel and initrd images
Overview - Kernel

- Kernel (as a compressed image file) is loaded into memory and decompressed.
- Looks for an init process to run

- Establishes memory management
- Switches to non-architecture specific Linux kernel functionality via start_kernel()
Overview - Init & Runlevel

- Init is responsible for starting all other processes
- Init establishes and operates the user space
- Runlevel takes a value from 0-6, determining which subsystems are to be made operational

→ BIOS next
BIOS

- Provides a series of routines to directly access hardware.
- Can be bypassed by programs which access hardware directly.
- BIOS routines invoked through a system of software interrupts.
- Stored in BIOS ROM (Read Only Memory), which is persistent storage (Doesn’t lose values when power is off)
CMOS Chip (Complementary Metal Oxide Semiconductor)

- Stores configuration regarding how OS is to be booted.
  - Sequence of devices to look at to boot the OS from.
  - Contains passwords, time and date info.
- Located on a separate battery powered CMOS RAM chip.
## CMOS Setup Example

### CPU Internal Core Speed
- **CPU Internal Core Speed**: 650MHz

### Boot Virus Detection
- **Boot Virus Detection**: Enabled

### Processor Serial Number
- **Processor Serial Number**: Disabled

### CPU Level 1 Cache
- **CPU Level 1 Cache**: Enabled

### CPU Level 2 Cache
- **CPU Level 2 Cache**: Enabled

### CPU Level 2 Cache ECC Check
- **CPU Level 2 Cache ECC Check**: Disabled

### BIOS Update
- **BIOS Update**: Enabled

### Quick Power On Self Test
- **Quick Power On Self Test**: Enabled

### HDD Sequence SCSI/IDE First
- **HDD Sequence SCSI/IDE First**: IDE

### Boot Sequence
- **Boot Sequence**: A,CDROM,C

### Boot Up Floppy Seek
- **Boot Up Floppy Seek**: Disabled

### Floppy Disk Access Control
- **Floppy Disk Access Control**: R/W

### IDE HDD Block Mode Sectors
- **IDE HDD Block Mode Sectors**: HDD MAX

### HDD S.M.A.R.T. capability
- **HDD S.M.A.R.T. capability**: Disabled

### PS/2 Mouse Function Control
- **PS/2 Mouse Function Control**: Auto

### OS/2 Onboard Memory > 64M
- **OS/2 Onboard Memory > 64M**: Disabled

### MPS 1.4 Support
- **MPS 1.4 Support**: Disabled

### PCI/VGA Palette Snoop
- **PCI/VGA Palette Snoop**: Disabled

### Video ROM BIOS Shadow
- **Video ROM BIOS Shadow**: Enabled

### C8000 — CBFFF Shadow
- **C8000 — CBFFF Shadow**: Disabled

### CC000 — CFFFF Shadow
- **CC000 — CFFFF Shadow**: Disabled

### D0000 — D3FFF Shadow
- **D0000 — D3FFF Shadow**: Disabled

### D4000 — D7FFF Shadow
- **D4000 — D7FFF Shadow**: Disabled

### D8000 — DBFFF Shadow
- **D8000 — DBFFF Shadow**: Disabled

### DC000 — DFFFF Shadow
- **DC000 — DFFFF Shadow**: Disabled

### Boot Up NumLock Status
- **Boot Up NumLock Status**: Off

### Typematic Rate Setting
- **Typematic Rate Setting**: Disabled

### Typematic Rate (Chars/Sec)
- **Typematic Rate (Chars/Sec)**: 6

### Typematic Delay (Msec)
- **Typematic Delay (Msec)**: 250

### Security Option
- **Security Option**: System

### ESC: Quit
- **ESC**: Quit

### F1: Help
- **F1**: Help

### PU/PD+/+-: Modify
- **PU/PD+/+-**: Modify

### F5: Old Values (Shift)F2: Color
- **F5**: Old Values (Shift)F2: Color

### F6: Load BIOS Defaults
- **F6**: Load BIOS Defaults

### F7: Load Setup Defaults
- **F7**: Load Setup Defaults
What happens when CPU is powered on?

• First instruction executed by the CPU is at xffffff.
  – Jump instruction to BIOS startup program entry point (inside BIOS ROM).
• BIOS startup program then runs the Power on Self Test (POST).
  – Full POST performed when the CPU is turned on.
  – If CPU is reset, some POST checks can be avoided.
POST

• Checks BIOS ROM and CMOS RAM chip for battery failure.
• Checks the integrity of various Hardware devices
  – Timer IC’s
  – Video ROM
  – Keyboard
  – DMA Controller
AFTER POST

- BIOS startup routine detects and executes BIOS device routines.
  - Used to initialize hardware (e.g. Video Mem)
- Perform memory test
- Configure PCI Bus devices.
- Assign IRQ numbers and other resources.
- Startup routine issues an int x19 to execute BIOS routine which locates OS.
Option ROMs

- Firmware called by system BIOS (BIOS extensions).
- BIOS routines specific to devices
  - Stored in a separate device-specific ROM
  - Can override the device routines provided by system BIOS
- Usually executed after POST
Option ROMs

• Video BIOS
  – Executed early on in boot process so POST results can be viewed.

• Network Boot ROM
  – Hooks onto BIOS routine x18, which is called when no bootable device is formed.
  – Allows computer to download OS.
Locating the OS

• BIOS refers to the sequence of devices in which the OS can be present from CMOS (Boot sequence).
• Bootable devices have to contain persistent storage.
• BIOS looks at the first sector (MBR) of a device to determine if device is bootable.
  – Last 2 bytes need to be x aa55
LOCATING the OS

• If device is Bootable:
  – BIOS loads MBR into RAM at linear address x7c00
  – Transfer control to MBR code.

• If device is not Bootable
  – BIOS examines the next persistent storage device from boot sequence.

→ bootloader next
Bootloader

• What does a bootloader do
  – Main purpose: Bring the kernel into memory
  – Provide the kernel with the information it needs to work correctly
  – Switch to an environment that the kernel will work properly
  – Transfer control to kernel
Bootloader

• Loading kernel to memory
  – code of bootloader knows where is the kernel and under what file system
  – kernel image reside in arch/i386/boot
  – if the kernel is modular, we need to load the modules along with the kernel
Bootloader

• Giving the kernel its information
  – For example, the root partition to start from, maybe a mapping of where physical memory is and where it’s not.
Bootloader Types

- **Single-Stage Bootloader**
  - consist of a single file (MBR), usually has limited size of 512 bytes

- **Two-Stage Bootloader**
  - Stage 1 bootloader
  - Stage 2 bootloader
Two-Stage Bootloader

• Why use Two-Stage Bootloader
  – MBR size is limited
  – Apart from the bootloader code, MBR contains BIOS structures and FAT file system headers, which leaves less spaces to work with
Two-Stage Bootloader

• Stage-1 Bootloader
  – Small
  – Sole purpose of loading the stage-2 bootloader

• Stage-2 Bootloader
  – Large
  – Contain all the code needed for loading the kernel and possible modules
Bootloader - GRUB 2

- GRand Unified Bootloader 2
- Example of Two-Stage bootloader
  - Rather a ‘Three-Stage’ bootloader
  - new stage 1.5: bootloader understand the
    particular file system containing the Linux
    kernel image.
  - can load linux kernel from an ext2 or ext3 file
    system
Bootloader - GRUB 2

• With the kernel, load the kernel image initrd (initial ramdisk) into memory

• When stage 2 loaded and executed
  – a list of available kernel is displayed
  – user select a kernel and can have additional kernel parameters
  – finally loads the selected kernel
Bootloader - GRUB 2

GNU GRUB  version 2.00-7ubuntu11

Ubuntu
Advanced options for Ubuntu
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)

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.

Picture from debuntu.org
Kernel phase

• What does a Kernel do
  – memory management
  – task scheduling
  – I/O
  – interprocess communication
  – Overall system control

• Kernel is loaded in two stages in Kernel phase
Kernel phase

```
start ()           ./arch/i386/boot/head.S
startup_32 ()      ./arch/i386/boot/compress/head.S
  decompress_kernel ()      ./arch/i386/boot/compress/misc.c
startup_32 ()      ./arch/i386/kernel/head.S
start_kernel ()    ./init/main.c
```

Picture from ibm.com
First Kernel Stage

• Kernel loading stage
  – starts via start() and startup_32() (for x86 based processors) process
  – 1st startup_32()
    • Initializes the segmentation registers and a provisional stack
    • Decompress the kernel via decompress_kernel()
    • Jumps to physical address 0x00100000, where the decompressed kernel image resides
Second Kernel Stage

- Kernel startup stage
  - 2nd startup_32():
    - Initializes the segmentation registers with their final values
    - Sets up the kernel mode stack for process 0
    - Puts the system parameters obtained from the BIOS and the parameters passed to the operating system into the first page frame
    - Identifies the model of the processor
    - Jump to start_kernel() function
Second Kernel Stage

• Kernel startup stage
  – start_kernel():
    • The page tables are initialized
    • The page descriptors are initialized
    • The system date and time are initialized
    • Setup interrupt handling (IRQ)
    • The kernel thread for process 1 is created. In turn, this kernel thread creates other kernel threads and executes the Init program.
Init

• The grandparent or parent of all processes that start up automatically on the Linux system

  – PID = 1
Init

• What does it do?
  – check the file systems
  – set the clock
  – initialize serial ports

  – set default run level between 0 to 6
  – ultimately switch to a user-environment when system startup is completed
Init

• Run level
  – a configuration of processes
  – how the system should be set up
  – Init reads /etc/inittab file for detail information for each run level
  – Level 5: Full Multiuser with Networking and X Windows (GUI)
Init

• After determines run level
  – init starts all of the background processes necessary for the system to run by looking in the appropriate rc directory for that specific run level
  – runs all of the start scripts in the appropriate run level directory so that all services and applications are started correctly.

→ Security & TPM next
# Booting Security

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIOS</td>
<td>Basic Input/Output System Executes MBR</td>
</tr>
<tr>
<td>MBR</td>
<td>Master Boot Record Executes GRUB</td>
</tr>
<tr>
<td>GRUB</td>
<td>Grand Unified Bootloader Executes Kernel</td>
</tr>
<tr>
<td>Kernel</td>
<td>Kernel Executes /sbin/init</td>
</tr>
<tr>
<td>Init</td>
<td>Init Executes runlevel programs</td>
</tr>
</tbody>
</table>
| Runlevel     | Runlevel programs are executed from /etc/rc.d/rc*.
d
Trusted Platform Module

What is TPM?
• Small cryptographic device
• Trusted boot
• Remote attestation
• Key management
TPM Components

- Non Volatile Storage (> 1280 bytes)
- PCR Registers (≥16 registers)
- Other Junk

Crypto Engine: RSA, SHA256, HMAC, RNG

credit: Stanford 2016
Non-Volatile Storage

• **Endorsement Key** (EK - 2048 bits)
  – Created at manufacturing
  – Used for attestation

• **Storage Root Key** (SRK - 2048 bits)
  – Used for encrypted storage

• **Owner Password** (160 bits) and other flags
Platform Configuration
Registers

• PCR specs
  – 32-byte SHA256 digests
  – At least 16 PCRs
  – Initialized as 0 when booted

• PCR extension with measurements
  – \(\text{PCR}[i] = \text{SHA}(\text{PCR}[i] || M)\)
Trusted Boot

- Measure booting components and extend PCRs
Remote Attestation

• Prove to server what code is loaded
  – e.g. Company requires employees to run a firewall program on their laptop

• Steps of Attestation
  – Client generates Attestation Identity Key, and send public key to attestation server
  – Server asks for signed PCR state
  – Check if PCR state match known good state
Key Management

• TPM can generate keys and encrypt them by SRK
  – encrypted key blobs are stored on hard disk
  – keys are loaded and decrypted at runtime in TPM
References

• http://www.webopedia.com/DidYouKnow/Hardware_Software/BootProcess.asp
• http://www.thegeekstuff.com/2011/02/linux-boot-process
• https://www.engineersgarage.com/tutorials/how-computer-pc-boots-up
• https://www.ibm.com/developerworks/library/l-bootload/
• http://www.tldp.org/LDP/intro-linux/html/sect_04_02.html
References

- [https://oxax.gitbooks.io/linux-insides/content/Booting/linux-bootstrap-1.html](https://oxax.gitbooks.io/linux-insides/content/Booting/linux-bootstrap-1.html)