

# CSGE602055 Operating Systems

## CSF2600505 Sistem Operasi

### Minggu 03: BIOS, Loader, Systemd, & I/O

Rahmat M. Samik-Ibrahim

Universitas Indonesia

<http://rms46.vlsm.org/2/207.html>

REV096 8-NOV-2017

|           |                      |  |
|-----------|----------------------|--|
| Minggu 00 | 29 Aug - 05 Sep 2017 | Intro & Review                                       |
| Minggu 01 | 07 Sep - 12 Sep 2017 | IPR, SED, AWK, REGEX, & Scripting                    |
| Minggu 02 | 14 Sep - 19 Sep 2017 | Protection, Security, Privacy,<br>& C-language       |
| Minggu 03 | 26 Sep - 30 Sep 2017 | BIOS, Loader, Systemd, & I/O                         |
| Minggu 04 | 03 Okt - 07 Okt 2017 | Addressing, Shared Lib, Pointer<br>& I/O Programming |
| Minggu 05 | 10 Okt - 14 Okt 2017 | Virtual Memory                                       |
| Ming. UTS | 15 Okt - 24 Okt 2017 |  |
| Minggu 06 | 26 Okt - 31 Okt 2017 | Concurrency: Processes & Threads                     |
| Minggu 07 | 02 Nov - 07 Nov 2017 | Synchronization                                      |
| Minggu 08 | 09 Nov - 14 Nov 2017 | Scheduling<br>& Network Sockets Programming          |
| Minggu 09 | 16 Nov - 21 Nov 2017 | File System & Persistent Storage                     |
| Minggu 10 | 23 Nov - 28 Nov 2017 | Special Topic: Retreat                               |
| Cadangan  | 30 Nov - 09 Des 2017 |  |
| Ming. UAS | 10 Des - 23 Des 2017 |  |

# Agenda

- 1 Start
- 2 Agenda
- 3 Week 03
- 4 Legacy BIOS
- 5 UEFI
- 6 UEFI Boot
- 7 Operating System (Boot) Loader
- 8 GRUB Map
- 9 init (SYSV legacy)
- 10 UpStart - Ubuntu
- 11 The All New "systemd"
- 12 systemctl
- 13 PCH: Platform Controller Hub
- 14 Some Terms
- 15 I/O
- 16 The End

# Week 03: BIOS, Boot, Systemd, and I/O

- Reference: (Any Related Tutorial) (ETC 300-324) (SUP WEEK03) (OLD 10)
- Firmware
  - BIOS: Basic Input Output System.
  - UEFI: Unified Extensible Firmware Interface.
  - ACPI: Advanced Configuration and Power Interface.
- Operating System (Boot) Loader
  - BOOTMGT: Windows Bootmanager / Bootloader.
  - LILO: Linux Loader.
  - GRUB: GRand Unified Bootloader.
- Operating System Initialization
  - Init (legacy)
  - UpStart
  - Systemd
- I/O
  - Interrupt.
  - DMA.
  - ETC.

- Check Settings.
- Initialize CPU & RAM.
- POST: Power-On Self-Test.
- Initialize ports, LANS, etc.
- Load a Boot Loader.
- Handover to the Boot Loader.
- Provides "Native" (obsolete) Drivers only (not loadable).
- Provides "INT" services .
- Limitation.
  - Technology of 1970s.
  - 16 bits software.
  - 20 bits address space (1 MB).
  - 31 bits disk space (2 TB).

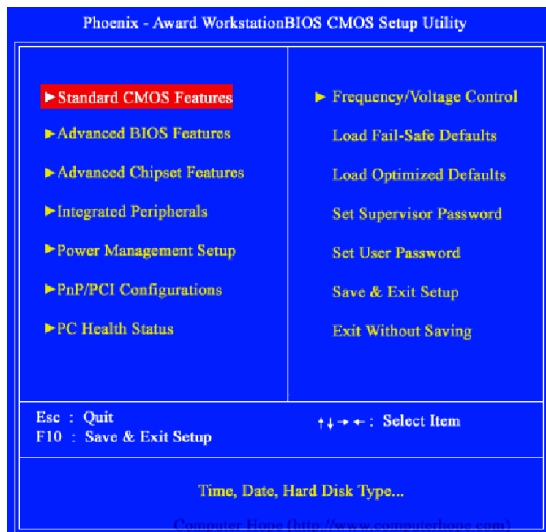


Figure: BIOS

- A Firmware Specification, not an Implementation!
- No (INT) service after boot.
- HII: Human Interface Infrastructure.
- Protected Mode.
- Flexible.
  - Technology of 2000s.
  - written in C.
  - (third party) loadable drivers and tools.
  - Emulate Legacy BIOS transition (MBR block, INT service).
  - UEFI Shell: environment shell for diagnostic (no need for DOS).
- Problems
  - Who controls the Hardware?
  - Is "Secure Boot" a good thing?
  - How about a **NASTY/LOCKING/TROJAN** UEFI implementation?
  - Different **DRIVERS**.



Figure: UEFI



## Platform Initialization (PI) Boot Phases



Figure: UEFI Boot Process<sup>1</sup>.

# Operating System (Boot) Loader

- General
  - How/Where to start the operating system?
  - What to do?
  - How many ways to boot?
  - How many types of OS?
- GRUB/GRUB2: GRand Unified Boot system
  - Stage 1 (boot.img): MBR (Master Boot Record) – Where is everything
  - Stage 1.5 (core.img): generated from diskboot.img
  - Stage 2: Kernel Selection: Windows, Linux, BSD, etc.
- GRUB2
  - More flexible than GRUB legacy
  - More automated than GRUB legacy
- Disk Partition
  - MBR: Master Boot Record (1983).
  - GPT: GUID Partition Table (2010s).

## GNU GRUB 2

Locations of *boot.img*, *core.img* and the */boot/grub* directory

Example 1: an MBR-partitioned harddisc with sector size of 512 or 4096Bytes



Example 2: a GPT-partitioned harddisc with sector size of 512 or 4096Bytes



Figure: GRUB<sup>1</sup>.

<sup>1</sup>Source Shmuel Csaba Otto Traian 2013

# init (SYSV legacy)

- File: `/etc/inittab`.
- Folders: `/etc/rcX.d` — `X` = runlevel.
  - Seven (7) different runlevels:
    - 0 (shutdown).
    - 1 (single-user/admin).
    - 2 (multi-user non net).
    - 3 (standard).
    - 4 (N/A).
    - 5 (3+GUI).
    - 6 (reboot).
  - `SXX-YYY`: Start
  - `KXX-YYY`: Kill.
- One script at a time in order.
- dependency is set manually.

- Developer: Ubuntu.
- Folder: `/etc/init/`.
- Control: `initctl`.
  - `initctl list` – listing all processes managed by upstart.
- better support for hotplug devices.
- cleaner service management.
- faster service management.
- asynchronous.

# The All New "systemd"

- Replaces (SYSV) init and UpStart.
  - better concurrency handling: Faster!
  - better dependencies handling: No more "S(tarts)" and "K(ills)".
  - better crash handling: automatic restart option.
  - better security: group protection from anyone including superusers.
  - simpler config files: reliable and clean scripts.
  - hotplug: dynamic start/stop.
  - supports legacy systems (init).
  - overhead reducing.
  - unified management way for all distros.
  - bloated: doing more with more resources.
  - linux specific: NOT portable.

# systemctl

```
for II in \
'systemctl list-unit-files | head -8; echo "(...)";
  systemctl list-unit-files| tail -8' \
'systemd-analyze blame | wc -l; echo "===";
  systemd-analyze blame | head -15' \
'systemctl --full | wc -l; echo "===";
  systemctl --full | head -10' \
'systemctl list-units | wc -l; echo "===";
  systemctl list-units | head -10' \
'systemctl list-units |grep .service|wc -l;echo "===";
  systemctl list-units|grep .service|head -10' \
'systemctl list-units | grep ssh.service' \
'systemctl status ssh.service' \
'systemctl is-enabled ssh' \
'journalctl' \
'journalctl -b' \
do
...
```

# PCH: Platform Controller Hub

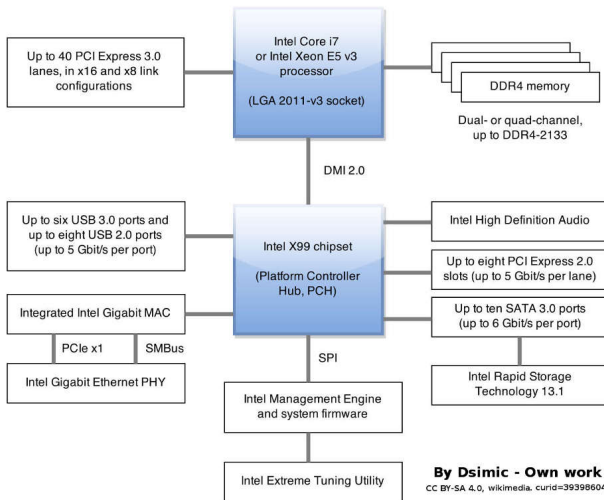


Figure: PCH: Platform Controller Hub



# Some Terms

- PCH: Platform Controller Hub
- PCIe: Peripheral Component Interconnect Express — 32 bits for (16 \* 1x or 8 \* 2x or 4 \* 4x or 2 \* 8x or 1 \* 16x) \* (2 direction) lanes.
- DMI: Direct Media Interface. Eg. DMI 2.0 (2 GB/s; 4x)
- GT/s: GigaTransfers per second
- 1 KB (KiloByte) = 1000 bytes — 1 KiB (Kibibyte) = 1024 bytes<sup>1</sup>
- SMB: System Management Bus
- SPI: Serial Peripheral Interface, a de facto standard bus.
- SATA: Serial AT Attachment. Eg. SATA 3.2  $\approx$  2 GB/s.
- DDR4 SDRAM: Double Data Rate Fourth-generation Synchronous Dynamic Random-Access Memory: 2 x DDR2 (DDR2 = 2 x DDR (DDR = 2 x SDRAM)). Eg. DDR4-3200 (8x SDRAM); Memory Clock: 400 MHz; Data Rate: 3200 MT/s; Module Name PC4-25600; Peak Transfer Rate: 25600 MB/s,

---

<sup>1</sup>In IT tradition; 1 KB = 1024 bytes

# I/O (1)

- Direct I/O vs. Memory Mapped I/O
- Interrupts: Non Maskable (NMI) vs Maskable (MI)
- DMA: Direct Memory Access
- I/O Structure:
  - Kernel (S/W).
  - I/O (S/W: Kernel Subsystem)
  - Driver (S/W)
  - Controller (H/W)
  - Device (H/W)
- I/O Streams
  - APP
  - HEAD
  - MODULES
  - DRIVER
  - H/W.

- I/O Interface Dimensions
  - Character-stream vs. Block;
  - Sequential vs. Random-access;
  - Sharable vs. Dedicated;
  - Parallel vs. Serial;
  - Speed;
  - Read Write – Read Only – Write Only.
  - Synchronous vs. Asynchronous;
  - Blocking vs. Non-Blocking.
- Where should a new algorithm be implemented?
  - APP?
  - Kenel?
  - Driver?
  - Controller?
  - HW?

# The End

- This is the end of the presentation.