Boot sequence of embedded Linux in a PC

Table of Contents

_ ▭ X

01 UEFI

Joel Fernandes

UEFI

- By definition the Unified Extensible Firmware Interface.
- Specifications that define architecture of the platform firmware used for booting and interface with the OS.
- Replacement to BIOS.

BIOS vs UEFI

	BIOS	UEFI
Real Mode	16 bit	32/64 bit
GUI	Basic visuals only keyboard	Mouse and keyboard with visually appealing interface
Interface Language	Assembly	C
Boot times	Slow	Faster
Partition size	2TB (MBR)	9.4ZB (GPT)
Security	Password security	Secure boot

UEFI Features

Services

Boot services and runtime services

Protocols

Communication between 2 binary modules. Similar to BIOS interrupt calls

Device drivers

Loaded from non-volatile memory to operate outside devices

UEFI system partition

Uses GPT. Special partition thats saves applications and their files

Graphics features

Uses GOP (Graphics Output Protocol). GUI based firmware interfaces

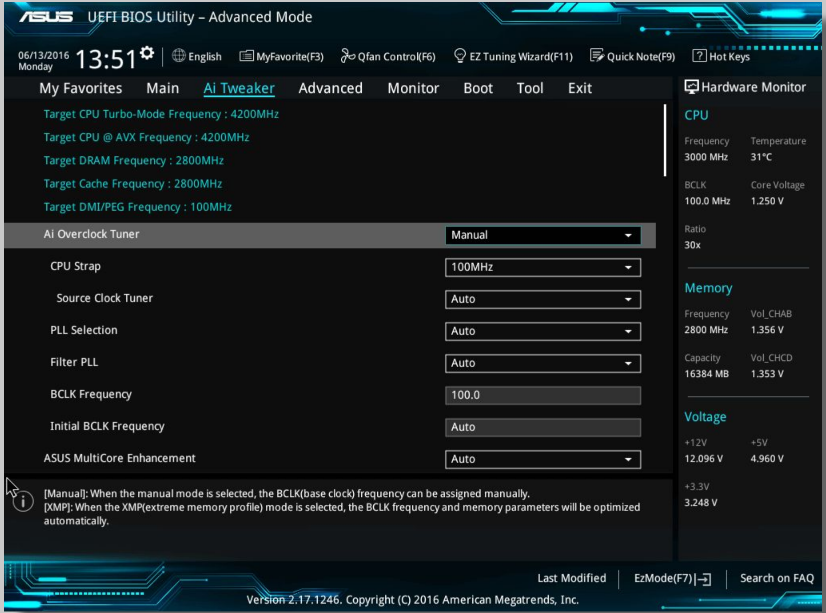
Hardware

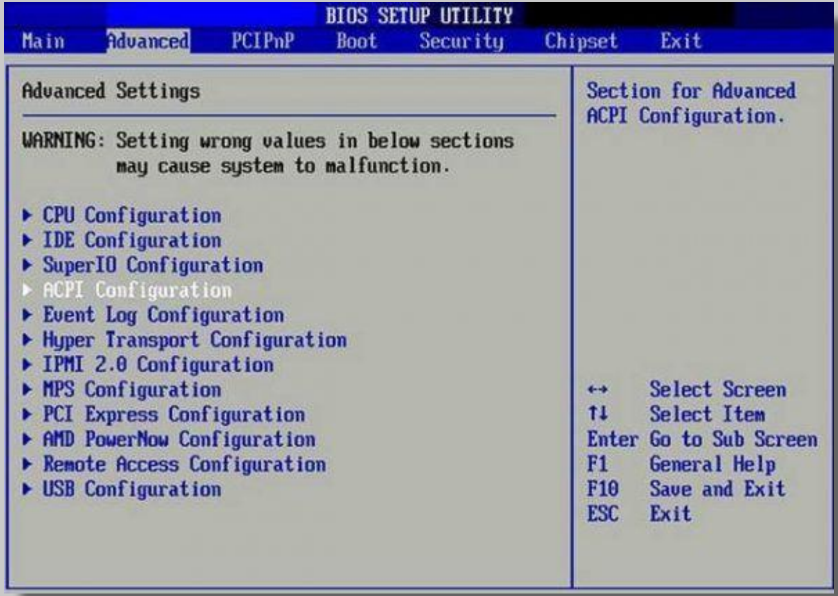
Tests hardware components before booting

Applications

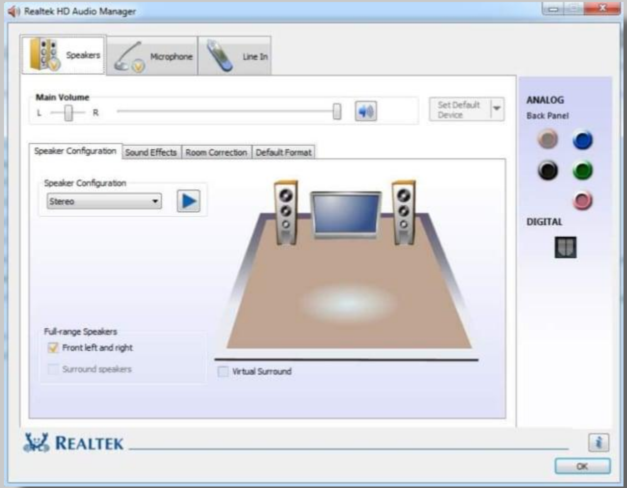
OS boot loader
UEFI shell

Examples of features





Examples of features



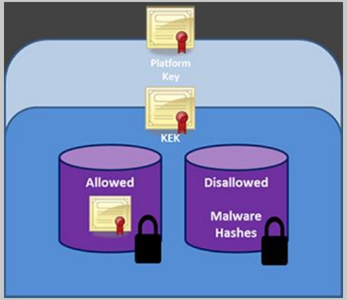
Examples of features

```
UEFI Interactive Shell v2.2
EDK II
UEFI v2.70 (EDK II, 0x00010000)
Mapping table
      FS0: Alias(s):HD0a0b::BLK1:
          PciRoot(0x0)/Pci(0x1E,0x0)/Pci(0x1,0x0)/Pci(0x5,0x0)/Scsi(0x0,0x0)/HD(
1,GPT,859897CF-0B19-4F0D-8A39-9E335DC964AC,0x800,0x100000)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x1E,0x0)/Pci(0x1,0x0)/Pci(0x5,0x0)/Scsi(0x0,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x1E,0x0)/Pci(0x1,0x0)/Pci(0x5,0x0)/Scsi(0x0,0x0)/HD(
2,GPT,7996C9DA-1FF2-4860-8AD9-130200E77EB6,0x100800,0x7A000)
     BLK3: Alias(s):
          PciRoot(0x0)/Pci(0x1E,0x0)/Pci(0x1,0x0)/Pci(0x5,0x0)/Scsi(0x0,0x0)/HD(
3,GPT,76EE77B1-C029-41C6-890F-19EA1F478E92,0x17A800,0x2685000)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
Shell> _
```

Adriano Soares

Secure Boot

Doesn't allow driver and bootloader execution with invalid digital signatures



02

Bootloader

Adriano Soares

What is a bootloader?

A small program that is responsible for placing a kernel image of an OS (or another bootloader) into RAM memory

Types of Bootloaders

First-stage bootloaders

BIOS (MBR), coreboot, Libreroot, Das U-boot…

Second-stage bootloaders

GNU GRUB, rEFInd, BOOTMGR, NTLDR, iBoot…

They are not OSs, they load an OS into RAM and transfer the control to its kernel!

Allow multiple OSs, multiple OS versions, OS initialization with different parameters…

Where are they located?

if (BIOS) return MBR;

In the 446 byte section of the Master Boot Record (512 bytes)

Not enough space to store large bootloaders with:

- Compatibility with complex and multiple filesystems
- Menu-driven selection of boot choices
- etc…

Solution: multiple stages - first stage loads the second stage, the second stage loads the kernel image of the OS

Problem: BIOS doesn’t know what a file system is, where’s the kernel image of the OS located?

Where are they located?

if (BIOS) return MBR;

<u>Problem</u>: where's the kernel image of the OS located?

<u>Solutions</u>:

1. Hardcoded sectors through *map* files - if either the kernel or the map files change location the *maps* and MBR must be updated, respectively

2. Make the bootloader file system aware (approach used by GNU GRUB)

Where are they located?

if (UEFI) return GPT;

UEFI is file system aware

Each OS has a “stage-1” bootloader in /EFI/boot with an .efi extension, that loads the “stage-2” bootloader in the corresponding DIR (e.g., /EFI/ubuntu)

```
/EFI
    /boot
    /microsoft
    /ubuntu
```

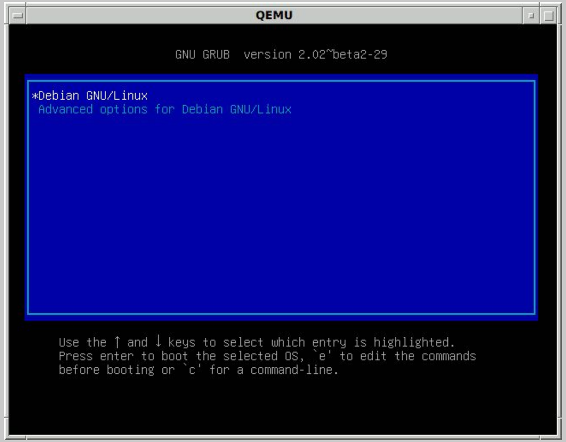
GNU GRUB

GRUB legacy

BIOS support

“2 stages”:

- stage-1 in MBR that loads stage-1.5
- stage-1.5 loads file system drivers, allowing to load stage-2 from /boot/grub
- stage-2 loads config files and other modules



GRUB2

BIOS and UEFI support

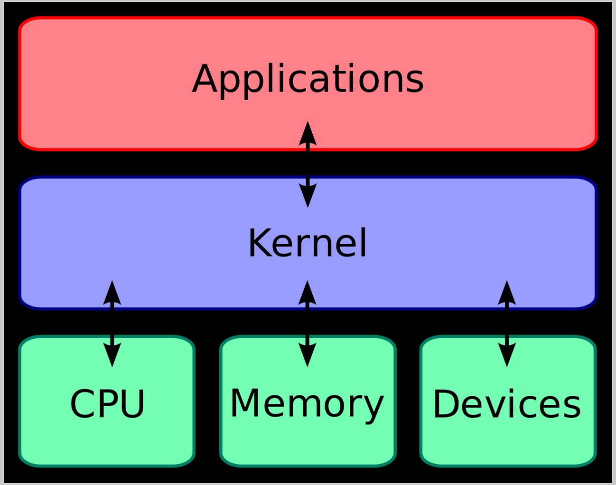
E.g. */EFI/ubuntu/grubx64.efi* that executes stage-2 directly, acting as stage-1 and stage-1.5

03 Kernel

Definition and Overview
Francisco Cerqueira

What is the Kernel?

- It's a program
- Interface between OS and hardware
- Performs crucial tasks
  - Process management
  - Memory management
  - File system control
  - Device control



Kernel Phase Overview

01

Decompression

Decompress itself in place

02

Verification

Perform hardware checks

03

Connection

Gain access to vital peripheral hardware

04

Initialization

Run the init process

Kernel Decompression

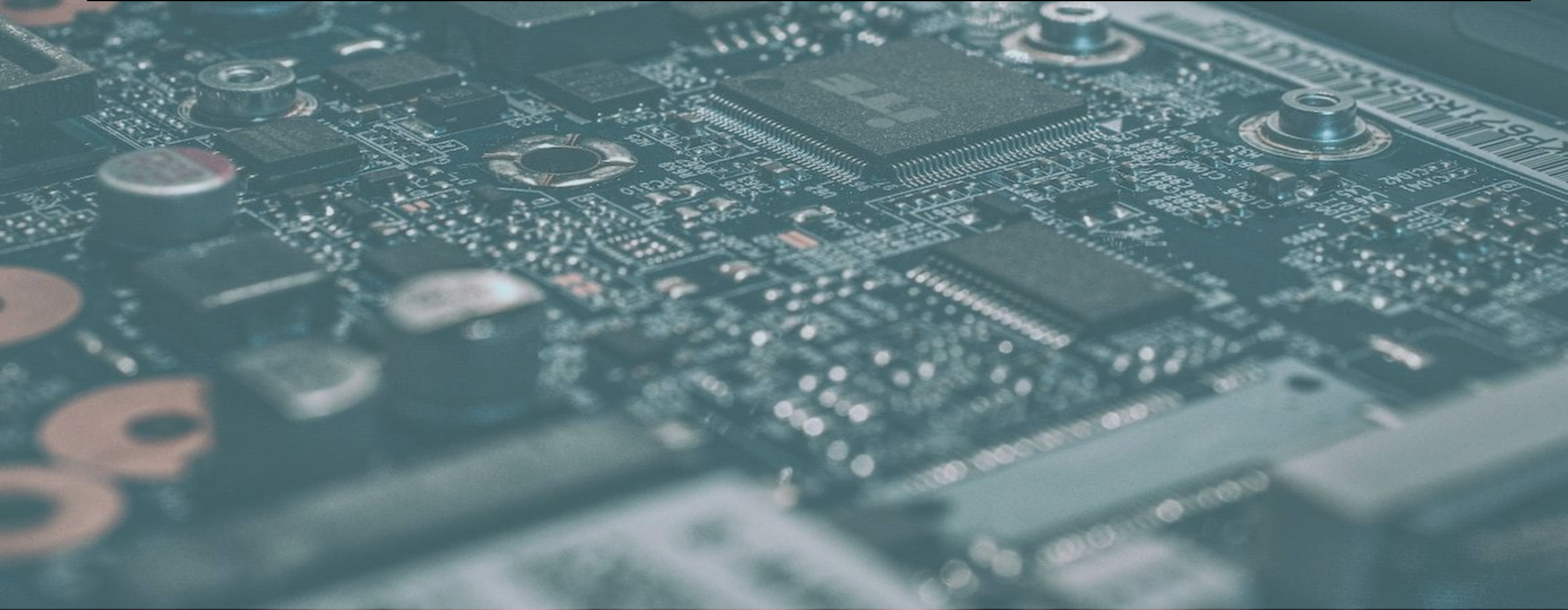
Compact Size

Saves space on the flash memory, as the compressed version is smaller.

Fast Performance

Most of the times it's faster to load and decompress than transfer a full kernel image from the storage to memory.

Hardware Verification



Peripheral Hardware Setup

Hardware Dependent

- Creation of page tables
- Initialization of:
    - Translation Look Aside Buffer
    - Cache
    - MMU

Hardware Independent

- Kernel Tick Control
- Run System Timer
- Setup a Stack per CPU
- Initialization of:
    - Memory Zones (DMA, High Memory, Low Memory)
    - Interrupt Tables
    - **Slab Allocation**
    - **/proc Filesystem**

Kernel Initialization

- System is now ready to call ***rest_init()***, creating the first process (PID 1).
- Initialization of:
  - Setup Network Socket Interface
  - Built in Drivers
- Privilege of execution set to **User mode**
- Initialization processes:
  - ***run_init_process("/sbin/init");***
  - ***run_init_process("/etc/init");***
  - ***run_init_process("/bin/init");***
  - ***run_init_process("/bin/sh");***

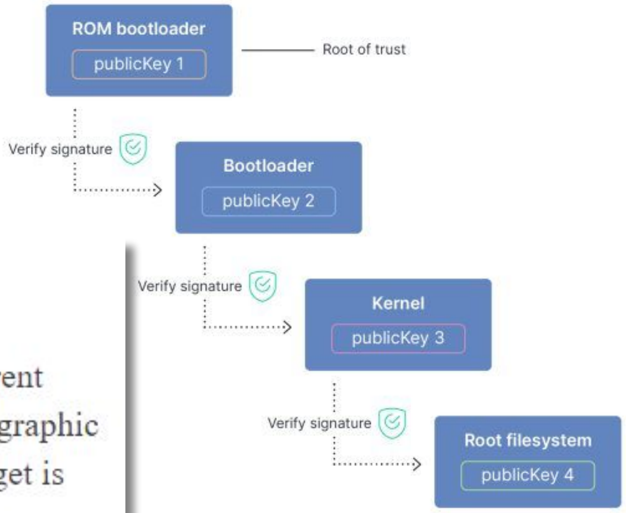
03 Kernel

Security - Eunice Amorim

Why is it important?

- Permission level high
- Partially operational
- Impact of the functionality

Chain of trust



dm-verity

Device-Mapper's "verity" target provides transparent integrity checking of block devices using a cryptographic digest provided by the kernel crypto API. This target is read-only.

Return Oriented Programing

- Buffer overflow attacks
- Reutilization of code with know position such as the kernel
- Smaller memories make the task easier

KASLR

01 Initialize dedicate page tables

02 Pages to avoid

03 Physical address randomization

04 Virtual address randomization

The process may have flaws...

VDB-222014 · CVE-2023-22997

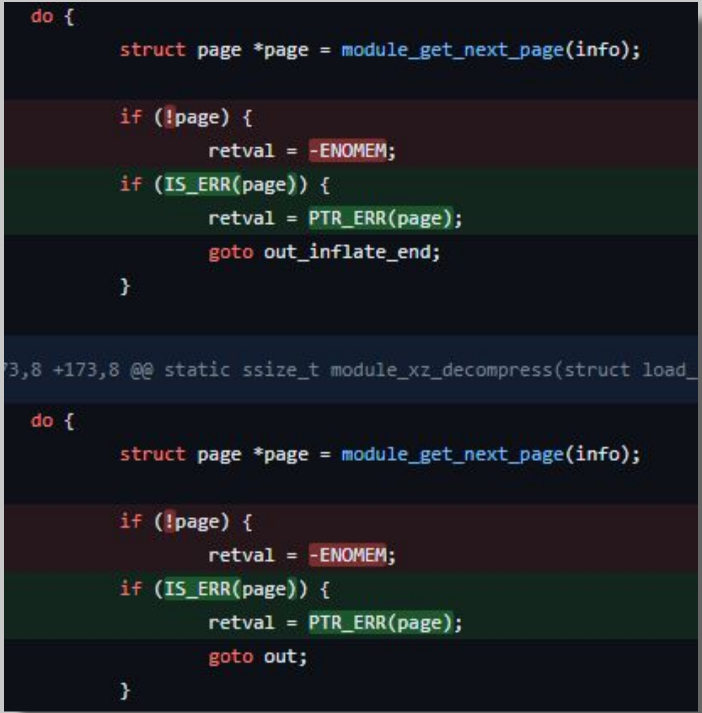
LINUX KERNEL DECOMPRESS.C MODULE_GZIP_DECOMPRESS MODULE_GET_NEXT_PAGE RETURN VALUE

ENTRY HISTORY DIFF JSON XML CTI

Linux

CVSS Meta Temp Score	Current Exploit Price (≈)	CTI Interest Score
5.4	$0-$5k	0.00

The process may have flaws...



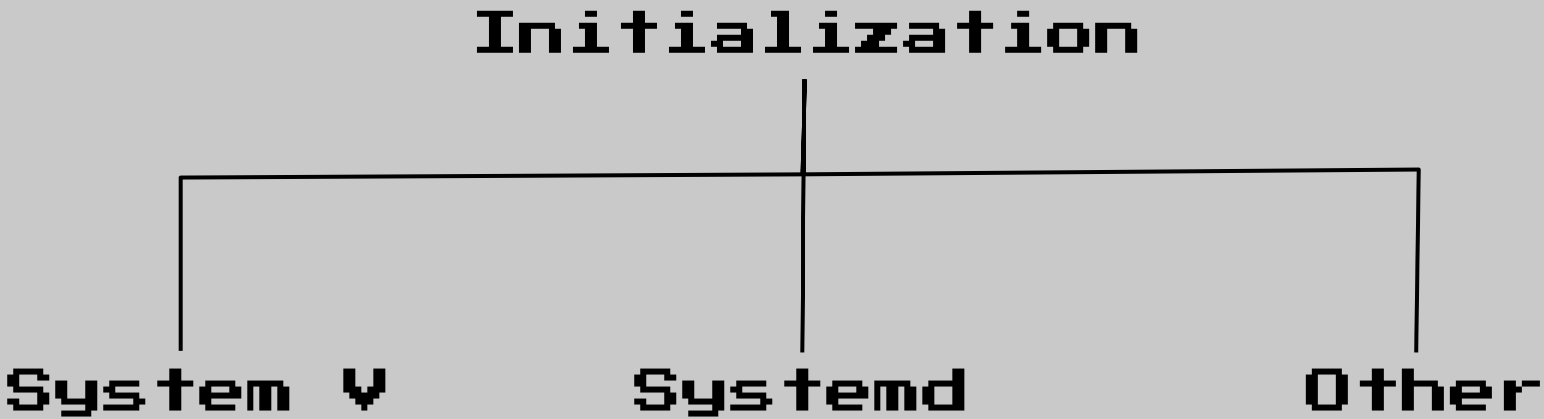
```
do {
        struct page *page = module_get_next_page(info);

        if (!page) {
                retval = -ENOMEM;
        if (IS_ERR(page)) {
                retval = PTR_ERR(page);
                goto out_inflate_end;
        }

3,8 +173,8 @@ static ssize_t module_xz_decompress(struct load_

do {
        struct page *page = module_get_next_page(info);

        if (!page) {
                retval = -ENOMEM;
        if (IS_ERR(page)) {
                retval = PTR_ERR(page);
                goto out;
        }
```

04

Operating System

Filipe Campos

System initialization



System V

Tasks:

- Verify the integrity of the local filesystems
- Mount local disks
- Designate and initialize paging areas
- Perform filesystem cleanup tasks
- Start daemons
- Enable user logins

System V - Runlevels

	Directory	Usage
N		System boot
0	/etc/r0.d/	Halt the system
1	/etc/rc1.d/	Single user mode
2	/etc/rc2.d/	Multi-user mode
3-5	/etc/rc3.d/ through /etc/rc5.d/	Identical to runlevel 2
6	/etc/rc6.d/	Reboot the system

Initialization scripts

- /etc/rcX.d/
- /etc/init.d/
- /etc/inittabs
- telinit <no>

```
lrwxrwxrwx 1 root root 16 Feb 16  2010 K10psacct -> ../init.d/psacct
lrwxrwxrwx 1 root root 15 Sep 10  2010 K15httpd -> ../init.d/httpd
lrwxrwxrwx 1 root root 13 Sep 10  2010 K20nfs -> ../init.d/nfs
lrwxrwxrwx 1 root root 14 Feb 16  2010 K25sshd -> ../init.d/sshd
lrwxrwxrwx 1 root root 17 Feb 16  2010 K30postfix -> ../init.d/postfix
lrwxrwxrwx 1 root root 19 Oct 12  2010 S20eventlogd -> ../init.d/eventlogd
lrwxrwxrwx 1 root root 16 Jan 28  2013 S21lsassd -> ../init.d/lsassd
lrwxrwxrwx 1 root root 20 Sep 10  2010 S22messagebus -> ../init.d/messagebus
```

Alternatives/Evolution

Upstart

Canonical's replacement for init.
Discontinued in 2014.

BusyBox init

Compact and more limited init mechanism for embedded systems.

Android

Accesses two boot scripts and includes the instantiation of a JavaVM

Other options

Launchd, Runit, Shepard, OpenRC, SystemStarted, Service Management Facility (Oracle)

SystemD

“When it comes to systemd, they expect me to have a lot of colorful opinions, but no. I don't personally care about systemd, in fact my main computer and laptop use it.”

— Linus Torvald

04

Operating System

SystemD - António Rente

SystemD

System and service manager, that runs as PID 1 and starts the rest of the system, intended as a SysVinit replacement.

History:

- Developed by Lennart Poettering & Kay Sievers, at Redhat (2010). First implemented in Fedora 14.

Features:

- Starts and manages *deamons* and processes using Linux control groups.
- Maintains mount points.
- Maintains a list system accounts and VMs.
- Manages network operations.
- Supports SysV scripts.
- Open source.

SystemD

“For a fast and efficient boot-up two things are crucial:
To start less. And to start more in parallel.”

[Rethinking PID 1]

– Lennart Poettering

SysVinit ≈ SystemD

Run level	Target
0	poweroff.target
1	rescue.target
2-4	multi-user.target
5	graphical.target
6	reboot.target

systemctl isolate runlevel5.target == systemctl isolate graphical.target

Main differences

Priority vs Dependencies

SysVinit requires manual prioritization of services to set an order of execution, SystemD is dependency based.

Easier parallelism

SystemD can start services in parallel with no restrictions, since dependency order is always satisfied. Less sh overhead.

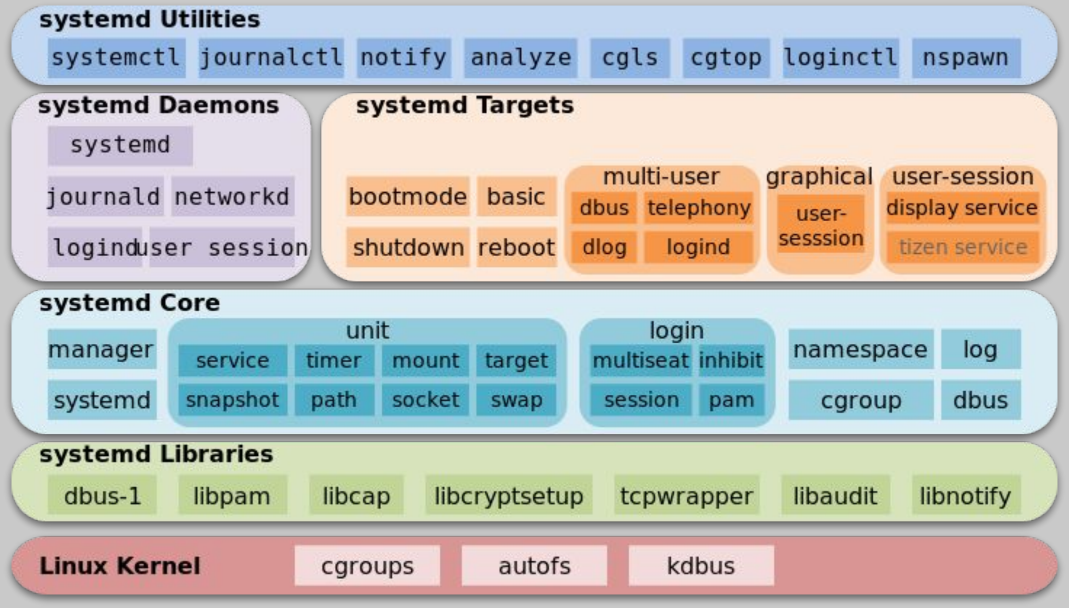
Start On Demand

Deamons are started once they are required (e.g. ssh).

Logs

SystemD configuration

- Units - ini-style config files (etc/systemd).
- Manage devices, sockets and services.
- Units' format defines their functionality: .device, .socket, .target, .service...
- systemctl for control and administration.



SystemD unit/commands

```
# /etc/systemd/system/redshift.service
[Unit]
Description=Redshift display colour temperature adjustment
Documentation="http://jonls.dk/redshift/"
After=graphical-session.target

[Service]
ExecStart=@bindir@/redshift
Restart=always

[Install]
WantedBy=graphical-session.target
```

- systemctl enable --now docker
- systemctl get-default
- systemctl list-dependencies graphical.target
- systemctl list-unit-files | grep enabled
- systemctl isolate <target>

Shortfalls of Systemd

Complexity

Systemd is a complex system with many components making it harder to understand and troubleshoot

Dependency

It has become a de facto standard in modern systems so many packages and tools now require it as a dependency

Larger attack surface

The larger codebase creates room for more security vulnerabilities compared to simpler options

Scope creep

Over time systemd has integrated more functionalities such as network management, journaling and containerization

Thank You

Known Vulnerability

CVE-2017-1000082 (CVSS Score 10.0)

“Systemd v233 and earlier fails to safely parse usernames starting with a numeric digit (e.g. "0day"), running the service in question with root privileges rather than the user intended.”