U1 – Chapter 2

Bjarte Kileng

HVL

January 9, 2018

Boot process and configuration

Select boot device:

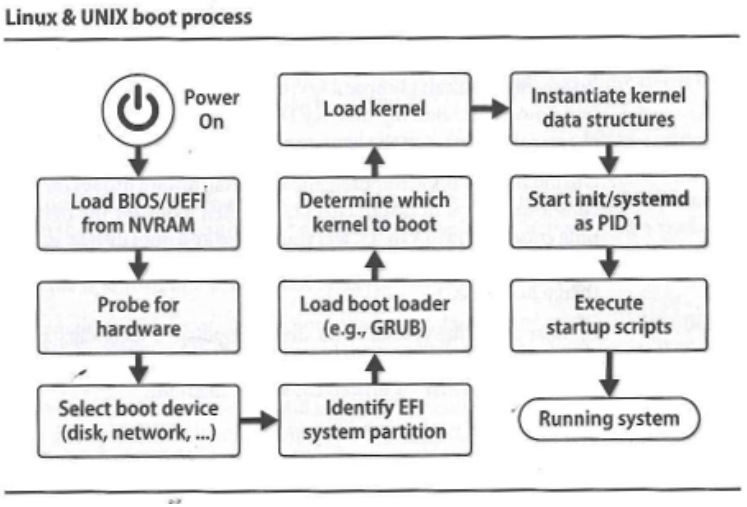
- Enable disk as boot device.
- Choose boot device.
- Specify boot device priorities.

Determine which kernel to boot:

- Select kernel to load..
- Specify default kernel.

Execute startup scripts:

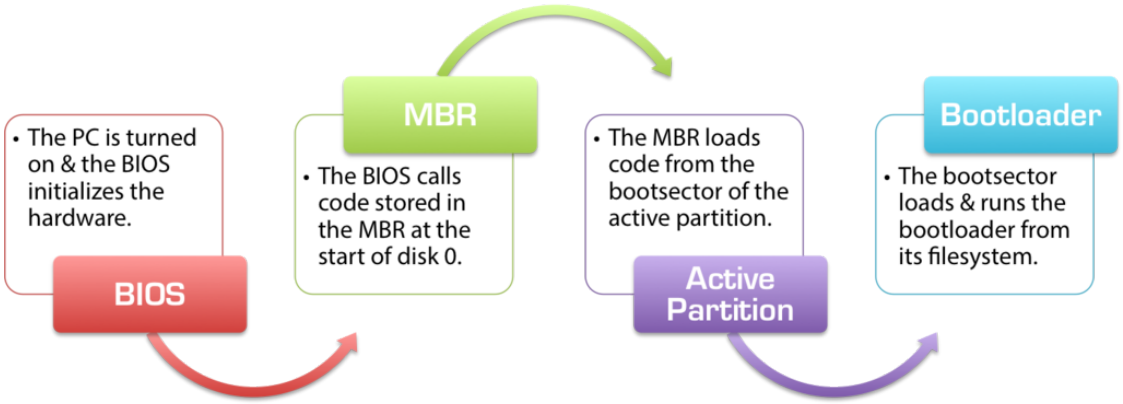
- Specify what init scripts to run.
- Run order or dependencies between init scripts.



BIOS and UEFI

- BIOS and UEFI are firmware that is run when the computer starts.
  - CPU is hardwired to run the firmware.
- Usually stored in EEPROM on the motherboard.
- Initializes the computer and runs the next stage of the bootstrapping code (e.g. GRUB).
- Typically includes a GUI that let us e.g. select the boot disk, or specify a disk priority for boot.
- UEFI has replaced BIOS on modern PCs.
- UEFI firmwares usually implement some kind of BIOS compatibility.
  - UEFI can boot from a MBR.

BIOS



- Boot disk must start with MBR that contains:
  - First-stage boot loader, *boot block*.
  - Disk partition table.
- The second-stage boot loader, e.g. GRUB:
  - Includes the necessary file system drivers to load the OS.
  - Loads the OS.
- The second-stage boot loader is read either from:
  - The active partition, or
  - the dead zone before the first partition (64 disk blocks).
- All BIOS configurations are done inside the firmware.
  - BIOS has no concept of OS or file systems.

UEFI acronyms

- UEFI = Unified Extensible Firmware Interface
- GUID = Globally Unique Identifier
- GPT = GUID Partition Table
- ESP = EFI System Partition
- FAT = File Allocation Table (MS_DOS file system)
- NVRAM = Non-Volatile RAM

UEFI

- UEFI firmware can handle FAT partitions.
- ESP is a FAT partition that stores the target application, e.g. GRUB.
  - Usually mounted as «/boot/efi».
- UEFI parameters are stored in NVRAM, accessible by UEFI and OS.
  - Accessible as «/sys/firmware/efi/efivars», or
  - «/sys/firmware/efi/vars».
- GPT identifies the ESP and the target application.
  - Stored as UEFI parameters.
  - Default target application is «/efi/boot/bootx64.efi».
- UEFI configurations can be modified in user space using e.g. the command **efibootmgr** or through the «/sys/firmware/» file system.

GRUB

- Default boot loader for Linux.
- Can boot multiple operating systems.
- GRUB code location:
  - UEFI – ESP
  - BIOS – Usually in the dead zone before the first partition.
- GRUB can read its configurations from a Linux file system.
  - Usually found as «/boot/grub2/grub.cfg».
- Modifications in «/boot/grub2/grub.cfg» do not survive updates.
  - Modify instead «/etc/default/grub».
  - Regenerate «/boot/grub2/grub.cfg» with the command:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

  - Before using *grub2-mkconfig*, «os-prober» must be installed.
- To specify a default kernel, see GRUB 2 - Fedora Project Wiki.

GRUB command line

- Let us modify the boot entries at boot time.
  - Changes are not saved.
- See the book or Internet for details.
- Forgotten the root password? Tell GRUB to start a bash shell:
  - Locate the menu entry.
  - Press the «e» key.
  - Add *init=/bin/bash* to the kernel line.
  - Use «Ctrl-x» to boot.

systemd

- Started by the kernel as process with pid equal to 1.
- Starts and stops system services and daemons.
- The systemd process is the top process in the process hierarchy.
  - The Linux kernel can namespace the process hierarchy.
  - Inside a process namespace, pid 1 is the first process of the namespace.

systemd and init

- The init system of System V is now replaced with systemd.
  - CentOS 7 is the first CentOS to use systemd.
- Startup sequence:
  - init – the scripts are run in a set sequence.
  - systemd – dependencies determine the startup sequence.
- Parallelism:
  - init – the scripts are run in a strict sequence, one at a time.
  - systemd – can run the scripts in parallel.
- systemd is more complicated system.

systemd units and unit files

- A unit is an entity that is managed by systemd:
  - Can be a service started by a script.
  - Can be a target (more later).
  - And much more.
- Every systemd unit must have a unit file:
  - Path of executable.
  - Specify how to start and stop.
  - Specify dependencies.
- With **systemd-run**, units can be started without a unit file.
  - Are named transient units.
- Unit files are read from:
  - «/etc/systemd/system» – Highest priority.
  - «/usr/lib/systemd/system»
  - «/lib/systemd/system»
- Unit-file suffix specify unit type:
  - E.g. «.service», «.target», «.timer», «.socket», «.mount», ...

Find systemd units

- List all loaded units:

```
systemctl list-units
```

- List loaded services only:

```
systemctl list-units --type=service
```

- List all service unit files:

```
systemctl list-unit-files --type=service
```

Enable/disable/mask

- An enabled service will be started at boot by systemd.

```
systemctl enable NetworkManager-wait-online.service
```

- A disabled service will not be started at boot by systemd.
  - Can still be started manually.
  - Can also be started due to a depending enabled service.

```
systemctl disable NetworkManager-wait-online.service
```

- A masked service can not be started.
  - Unit file is linked to «/dev/null».

```
systemctl mask NetworkManager-wait-online.service
```

Start/stop/status

- A service can be started.

```
systemctl start chronyd.service
```

- A service can be stopped.

```
systemctl stop chronyd.service
```

- We can check the status of a service.

```
systemctl status -l chronyd.service
```

or

```
systemctl status -l --no-pager chronyd.service
```

Enablement state

- Only unit files with an `Install` section can be enabled or disabled with **systemctl**.
- The **list-unit-files** sub command list service and enablement state.
  - See the man page for all the enablement states.
- Static services can only be started by hand, or if named as a dependency by an enabled service.

targets

- Group together unit files.
- List loaded targets:

```
systemctl list-units --type=target
```

- List all targets:

```
systemctl list-unit-files --type=target
```

- Some targets have special meaning.
  - Run automatically by system at specific events.

Some special targets

ctrl-alt-del.target: Run when Control+Alt+Del is pressed.

default.target: Default unit to start at boot.

emergency.target: Run if boot fails, e.g. due to a failing local disk mount.

- Play with this mode **before** you need it.

graphical.target: Sets up the graphical login screen.

multi-user.target: Sets up a non-gui multi-user system.

reboot.target: Shutdown and reboot the system.

shutdown.target: Shutdown the system.

rescue.target: Pulls in the base system for administrative purpose.

Using targets

- The sub command **isolate** changes the current mode:

```
systemctl isolate multi-user.target
```

  - Activate the target with its dependencies, and deactivate all others.
- Show the default target:

```
systemctl get-default
```

- Set the default target:

```
systemctl set-default multi-user.target
```

- Can also specify a target on the kernel boot line
  - Emergency mode: `systemd.unit=emergency.target`
  - Rescue mode: `systemd.unit=rescue.target`.

Unit dependencies

- Dependencies are specified in the unit files.
- Keywords are e.g. *Wants*, *WantedBy*, *Requires*, *RequiredBy*, *Conflicts*.
  - See the book and manual pages for more keywords and their meaning.
  - Does not imply any sequence for processing.
- Keywords to serialize processing are e.g. *After*, *Before*.
  - If no serializing keyword, systemd will try to run processes in parallel.
- Systemd assumes a default set of dependencies for units.
  - Turn off assumption with `DefaultDependencies=false`

Unit file

- New unit files can be created in an editor.
- The systemctl sub command **edit** lets us modify an existing unit-file.

```
systemctl edit httpd.service
```

  - Creates an override in «/etc/systemd/system/httpd.service.d/».
- Do not modify existing unit-files.
  - Use an override, e.g. «/etc/systemd/system/httpd.service.d/override.conf».
- Using unit-file overrides:
  - The original and the override will be merged by systemd at use.
  - If collisions, the override has higher priority.
- Install section can not be modified by an override.

Some words on logging

- Systemd has its own logging system, manged by the *journald* daemon.
- Systemd log messages are stored in the «/run» directory.
  - Typically, the rsyslog daemon will process also the systemd logs.
- Can be accessed with the command **journalctl**.
  - Show log of the Bluetooth unit:

```
journalctl -u bluetooth.service
```

- System logging will be covered later.