Embedded Linux Device Drivers

OS Booting

- Power ON
- Firmware program --> RAM
- POST -- check all peripherals
- Bootstrap loader -- find bootable device/partition
- start bootloader -- get OS to boot from user (if multiple)
- start bootstrap program
- load kernel

Linux Booting (on PC -- GrUB)

- Power ON.
- Firmware programs loaded into RAM.
- Run POST -- check all peripherals.
- Run Bootstrap Loader -- Find the bootable device (Linux GrUB loader).
- Run GrUB stage 1.
    - First 512 bytes of the device.
    - Transfer control to GrUB stage 1.5.
- Run GrUB stage 1.5.
    - Between boot sector of the device and the First partition of the disk.
    - Have Linux FS driver (to read files from Linux partition).
    - Transfer control to GrUB stage 2.
- Run GrUB stage 2.
    - On Linux boot partition.
    - Reads /boot/grub/grub.cfg file.

```
menuentry 'Ubuntu' --class ubuntu --class os {
  linux /boot/vmlinuz-5.15.0-76-generic root=/dev/sda1 ro
    initrd  /boot/initrd.img-5.15.0-76-generic
}
```

    - Present options to end user.
- End user selects Linux kernel and Initial RAM fs to boot from and GrUB loads them into RAM.
- The kernel get self-extracted and execution of the kernel begins.
- In initrd (initial ram disk -- loaded into the RAM by GrUB) serves as temporary filesystem for Linux kernel until root filesystem from disk is available. It contains device drivers for minimal required devices like disk, ...
- Kernel use initrd drivers to access "root filesystem" and then access all files/drivers there. Now kernel starts first user space process i.e. init/systemd.
- The init/systemd process start the kernel booting sequence. Kernel booting is logically divided into runlevels/init levels/systemd targets.
    - 1 (rescue.target) - Single user (root login)
    - 2 (multi-user.target) - Multi user (no network)
    - 3 (multi-user.target) - Network (no GUI)
    - 4 - Reserved
    - 5 (graphical.target) - GUI

Embedded Linux

- Prime components of Embedded Linux
    - Embedded Hardware
    - Bootloader
    - Kernel
    - Root Filesystem/Busybox
    - Application/Drivers
- Steps to build Embedded Linux 0. Host system (PC) with Ubuntu/Debian
    1. Install toolchain and necessary tools
    2. Download the source code of Linux Kernel, Bootloader (u-Boot), BusyBox
    3. Build Bootloader
    4. Build Linux Kernel

5. Build RootFS with BusyBox
6. Prepare Boot media (SD-Card BOOT and ROOT partition)
7. Boot the Embedded system (board)