3.4.3 LINUX DEVICE TREE

A **Device Tree** is a data structure used by the **Linux kernel** to describe the **hardware components of a system** (especially embedded systems) that **cannot be detected automatically**.

Definition:

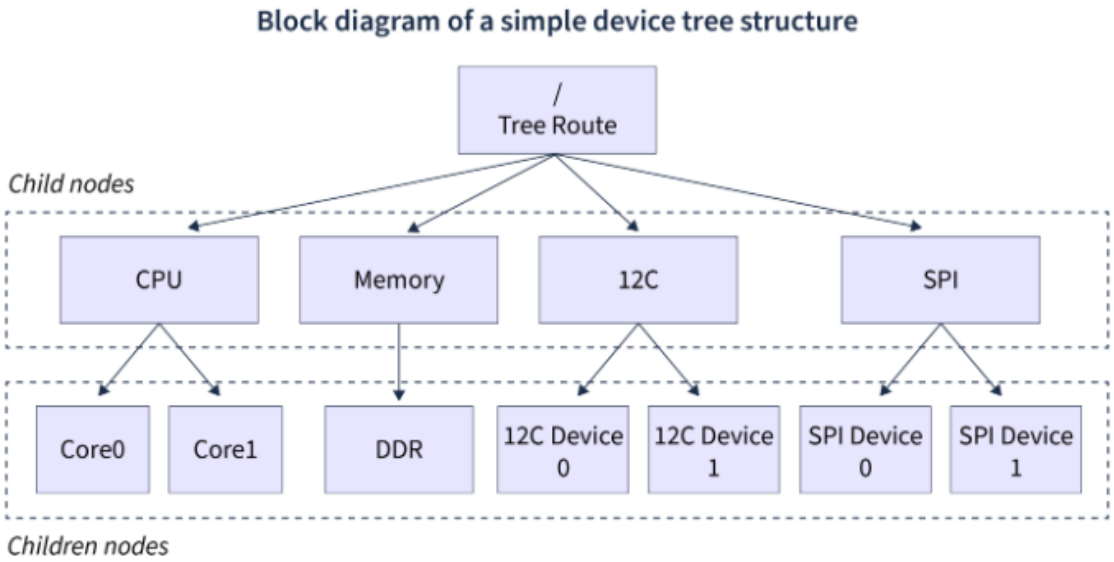
A **Device Tree** is a **hardware description** file that tells the Linux kernel what hardware is present on the system and how it is connected. A .dts file describes CPU, memory, UART, I²C devices, bootargs.

Need for Device Tree

- In embedded systems and Linux, the kernel cannot always automatically detect hardware.
- A **Device Tree (DT)** provides a **data structure** (in .dts files) describing:
    - What devices exist.
    - Their connections (**how to communicate** with them).
    - Manages drivers and resource allocation.
    - Hardware properties (e.g., addresses, interrupts, configurations).
- Example: Instead of hardcoding that the system has "2 CPU cores, DDR memory, I2C devices," we write it in the **device tree**. At boot, the kernel reads it and configures the hardware properly.

Device Tree Contains

- CPUs and memory
- Interrupt controllers
- Buses (I2C, SPI, USB)
- Peripherals (e.g., GPIO, UART, display, storage)
- Address mappings and IRQs
- Clocks, regulators, and other hardware resources



The above diagram is a **Block diagram of a simple device tree structure**.

1. Tree Root (/)

- At the very top, there is a single root node (/), called the **Tree Root**.
- This is like the starting point of the device tree, similar to the root directory in a filesystem.
- Describes the system as a whole.
- Contains **global properties**.

2. Child Nodes

- From the root node, we have **child nodes** that represent the main hardware blocks of the system:
  - **CPU -** Lists CPU cores and their properties.
  - **Memory -** Describes system RAM location and size.
  - **I²C (Inter-Integrated Circuit) –** it is a **two-wire serial bus** (SDA + SCL).
    In DT, which describe the **I²C controller (master)** and the **devices (slaves)** connected to it. It is one of the peripheral devices like UART, GPIO,SPI, Ethernet, USB, etc
  - **SPI -** SPI (Serial Peripheral Interface) is a **four-wire bus** (MISO, MOSI, CLK, CS).
    In DT, we describe the **SPI controller (master)** and the **devices (slaves)** with their chip select lines.

These are the primary components directly connected to the system root.

3. Children Nodes (Sub-nodes)

- Each child node can have its own **children nodes** (sub-devices or sub-components):
  - **CPU**
    - Has **Core0** and **Core1** (two processor cores).
  - **Memory**
    - Has **DDR** (a type of memory).
  - **I2C** (Inter-Integrated Circuit bus)
    - Has **I2C Device 0** and **I2C Device 1** attached.
  - **SPI** (Serial Peripheral Interface bus)
    - Has **SPI Device 0** and **SPI Device 1** attached.

4. Other components

- The structure shows a **hierarchical representation of hardware** in a system.
- Just like a **family tree**, parent nodes represent buses/controllers, while children represent devices connected to them.
- This hierarchy is **used by the operating system kernel** (like Linux) to understand what hardware is present and how it is connected.

SoC (System on Chip) / Bus

- Groups devices connected to the SoC.
- Defines buses like I2C, SPI, PCIe, etc.

Clocks, Resets, and Power Domains

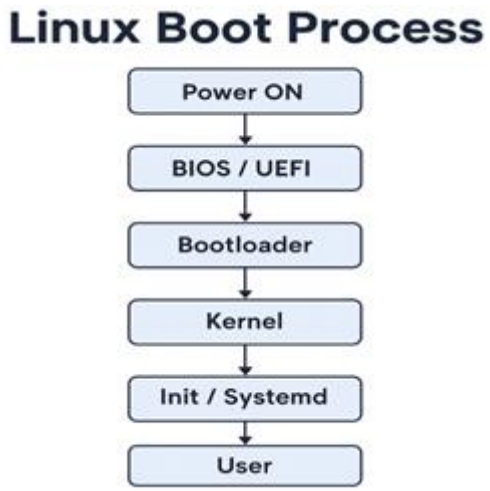
- Describe clock trees, reset lines, and power regulators.

Aliases

- Provide easy names for nodes.

Files Structure of Device Tree

1. **.dts – Device Tree Source**
   Human-readable text file describing hardware.
2. **.dtsi – Device Tree Source Include**
   Reusable parts of device trees (like SoC definitions).
3. **.dtb – Device Tree Blob**
   Compiled binary form of .dts loaded by the bootloader (e.g., U-Boot).



This diagram shows the **Linux Boot Process** – the sequence of steps that occur from when a computer is powered on until the user gets control of the system.

1. Power ON

- When the system is powered on, electricity flows into the hardware.
- The CPU starts executing the first instruction from a predefined location (usually stored in firmware).
- This initializes the system hardware.

2. BIOS / UEFI

- **BIOS (Basic Input/Output System)** or **UEFI (Unified Extensible Firmware Interface)** is the first software that runs.
- It performs **POST (Power-On Self Test)** to check hardware (RAM, CPU, keyboard, storage).

- It then locates a bootable device (HDD, SSD, USB, network).
- Finally, it loads the **bootloader** from the boot device into memory.

3. Bootloader

- A small program (e.g., **GRUB, LILO, systemd-boot**) responsible for loading the Linux kernel.
- Functions:
    - Presents a boot menu (if multiple OSes are installed).
    - Loads the selected kernel and **initramfs** (initial RAM filesystem).
    - Passes control to the kernel.

4. Kernel

- The **Linux kernel** initializes:
    - Device drivers
    - Memory management
    - Process scheduling
    - Mounts the **root filesystem**.
- After setup, the kernel starts the **Init process** (or systemd in modern distros).

5. Init / Systemd

- **Init** (older systems) or **Systemd** (modern systems) is the **first user-space process** (PID 1).
- Responsible for:
    - Launching background services (daemons) such as networking, logging, etc.
    - Setting the runlevel/target (multi-user, graphical, etc.).

6. User

- After services start, the system provides:
    - **Login prompt** (CLI or GUI).
    - User can now log in and interact with the OS.

Device Management Role

Device Management Task	Role of Device Tree
Device Detection	Lists hardware that can't self-report
Driver Binding	Helps kernel match drivers with hardware
Resource Management	Describes memory, IRQs, addresses
Power Management	Defines regulators, power domains
System Configuration	Custom configurations for different board versions

Advantages of Device Trees

- Separates **hardware description from kernel code**
- Supports **multiple board versions** without recompiling the kernel
- Simplifies **porting Linux to new hardware**
- Allows **flexible configuration at boot time**

Commands & Tools

Tool/Command	Description
dtc	Device Tree Compiler (.dts ↔ .dtb)
/proc/device-tree/	Runtime view of loaded Device Tree
fdtget, fdtdump	Inspect/manipulate .dtb files

Applications of Linux Device Tree:

1. Board bring-up
2. Hardware abstraction
3. Kernel portability
4. Dynamic hardware configuration
5. Virtualization (QEMU, KVM)
6. IoT & embedded devices (Raspberry Pi, phones, routers)