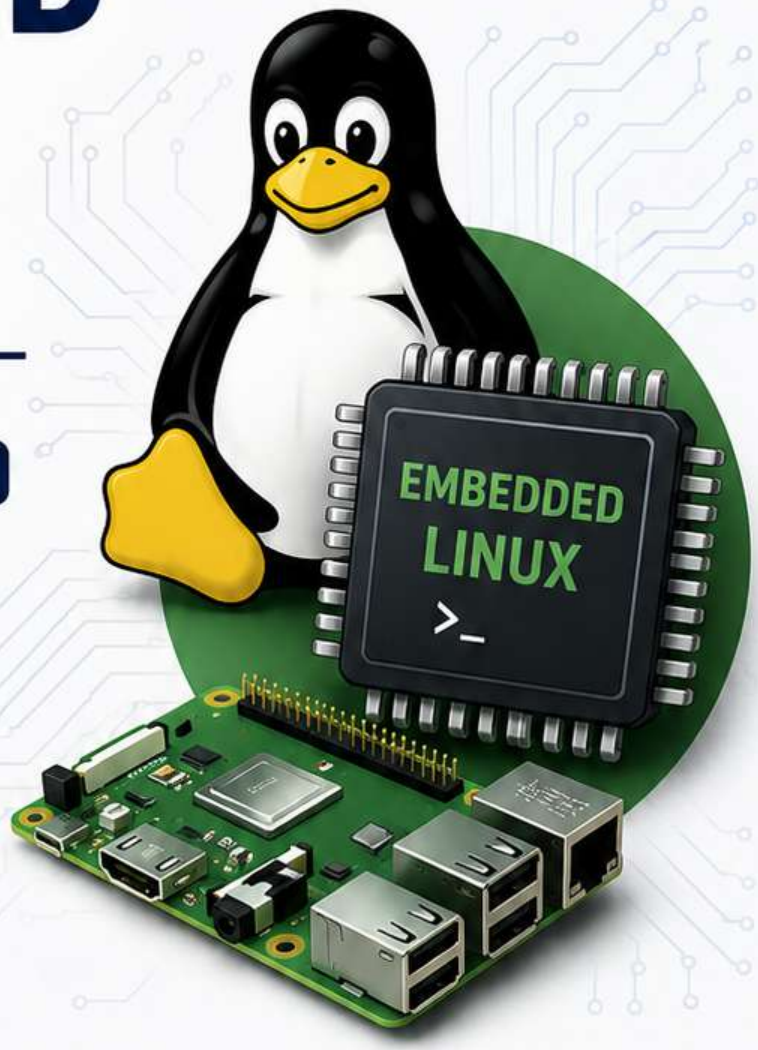
EMBEDDED SYSTEMS NOTES

LINUX

EMBEDDED LINUX NOTES

COMPLETE GUIDE FOR BEGINNERS

Learn | Understand | Implement



INSIDE THIS NOTES

- Linux Kernel
- Bootloader (U-Boot)
- Root File System
- Device Drivers
- Device Tree
- Build System (Buildroot & Yocto)
- Essential Commands
- Interview Questions

FOR EMBEDDED DEVELOPERS | PRACTICAL EXAMPLES | COMMANDS & TOOLS | REAL WORLD APPLICATIONS

ABOUT EMBEDDED LINUX

Embedded Linux brings the power and flexibility of Linux to embedded systems. It provides a stable, secure and customizable platform for modern embedded applications.

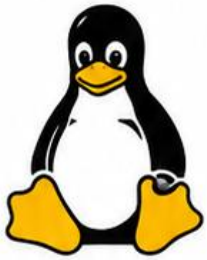
USED IN

Raspberry Pi | BeagleBone Black | Android Devices | Network Devices | Automotive Systems | Smart Devices

LINUX KERNEL

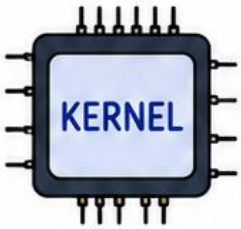
1 Q1. What is Linux Kernel?

Kernel is the core part of Linux that manages hardware and software communication.



2 Q2. What are the main functions of Kernel?

- ✓ Process Management
- ✓ Device Management
- ✓ Memory Management
- ✓ File System Management



3 Q3. Why is Kernel important in Embedded Linux?

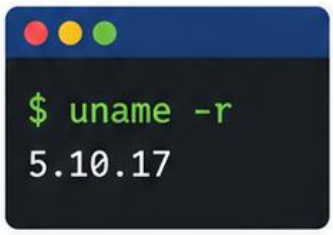
It controls hardware resources and allows applications to interact with devices.



4 Q4. How to check Kernel Version?

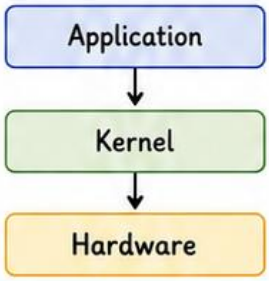
```
uname -r
```

Example Output: `5.10.17`



5 Q5. Does application directly access hardware?

No. Applications access hardware through the Kernel.



6 Q6. Name some devices using Linux Kernel.

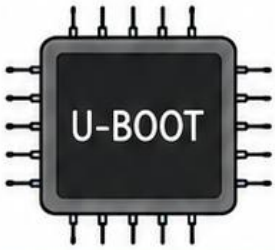
- Raspberry Pi
- BeagleBone Black
- Smart TV
- Router



BOOTLOADER (U-BOOT)

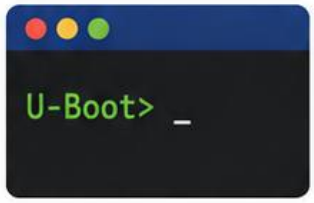
1 Q1. What is a Bootloader?

A Bootloader is a special program that runs first when the device powers on and loads the operating system (Linux Kernel) into RAM.

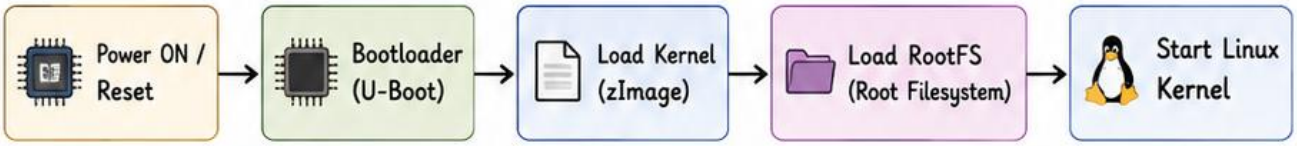


2 Q2. What are the main functions of Bootloader?

- ✓ Initialize hardware
- ✓ Provide interface (CLI)
- ✓ Load Kernel into RAM
- ✓ Load Device Tree (DTB)
- ✓ Load Root Filesystem
- ✓ Start the OS



3 Q3. What happens when device is powered ON?



4 Q4. How to check U-Boot version?

```
U-Boot> version
```

Example Output:

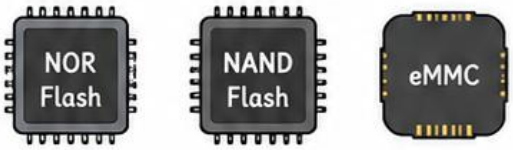
```
U-Boot 2021.04 (Jan 01 2024 - 10:30:00 +0000)
CPU:   ARMv7
Board: BeagleBone Black
DRAM:  512 MB
```

Note:

U-Boot command interface is used to configure and boot the system.

5 Q5. Where is Bootloader stored?

Bootloader is stored in non-volatile memory like NOR Flash, NAND Flash or eMMC.



6 Q6. Common U-Boot commands

Command	Description
help	Show all available commands
printenv	Print environment variables
setenv <var> <value>	Set environment variable
saveenv	Save environment variables
boot	Boot the kernel

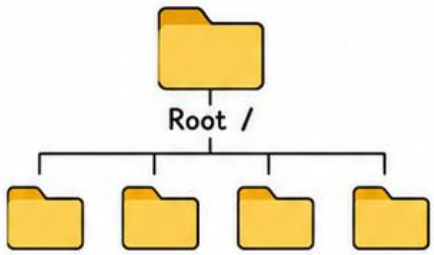
Tip:

Environment variables in U-Boot control booting process.

ROOT FILE SYSTEM (ROOTFS)

1 Q1. What is Root File System (RootFS)?

Root File System is the top-level directory structure in Linux. It contains essential binaries, libraries, configuration files and device nodes required to boot and run the system.

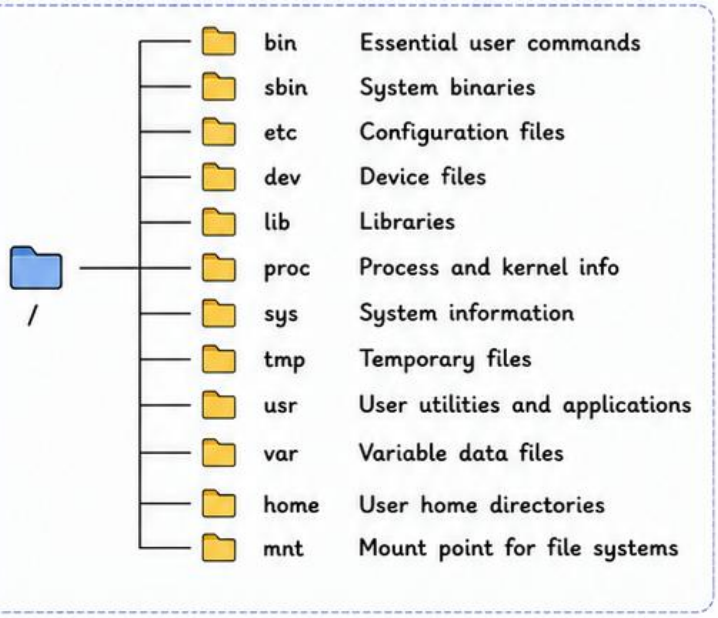


2 Q2. What are the characteristics of RootFS?

- It is the primary file system mounted at boot.
- Provides basic directory structure and essential files.
- Stored in flash memory in embedded systems.
- Read-only or Read-write (depends on system).

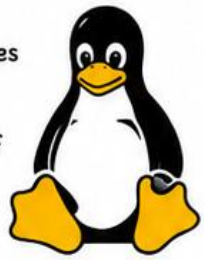
Stored in
NAND Flash
eMMC
SD Card

3 Q3. Show the basic directory structure of RootFS.



4 Q4. Why is RootFS important in Embedded Linux?

- Without RootFS, the system cannot boot.
- It provides all necessary files and directories.
- Ensures proper execution of applications and services.
- Lightweight RootFS saves memory and storage.



5 Q5. Where is RootFS mounted?

RootFS is mounted at the root directory "/" during system boot.

```
mount | grep " o
/
/dev/root on / type ext4 (rw)
```

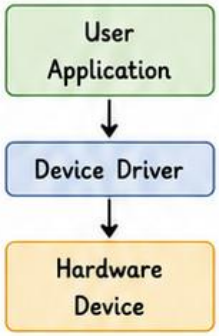
6 Q6. What are common types of RootFS used?

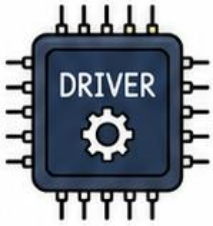
JFFS2	YAFFS2	UBIFS	EXT2/3/4	RAMFS	SQUASHFS
Used with NOR/NAND Flash	Used with NAND Flash	Modern NAND Flash File System	Used with NAND, eMMC, SD Card	Resides in RAM (Temporary)	Compressed Read-only File System

DEVICE DRIVER

1 Q1. What is a Device Driver?

A device driver is a software program that allows the operating system (Linux) to communicate with a hardware device.





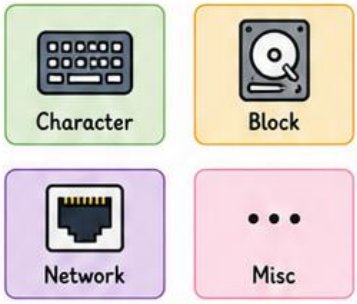
2 Q2. Why are Device Drivers Important?

- ✓ Enables OS to control hardware devices.
- ✓ Provides abstraction between hardware and software.
- ✓ Allows hardware independent programming.
- ✓ Improves system stability and performance.



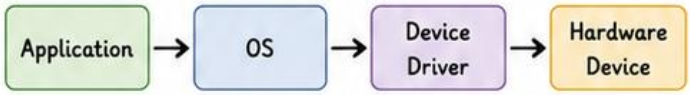
3 Q3. Types of Device Drivers in Linux

- **Character Device Drivers** – Handle data as a stream of bytes.
- **Block Device Drivers** – Handle data in blocks (e.g., storage).
- **Network Device Drivers** – Handle network interfaces.
- **Miscellaneous Drivers** – For devices that don't fit above.



4 Q4. How does a Device Driver Work?

1. Application sends a request to the OS.
2. OS calls the appropriate device driver.
3. Driver communicates with the hardware.
4. Driver returns the response to the OS.



5 Q5. How to see loaded drivers in Linux?

Use the following command:

```
lsmod
```

Example Output:

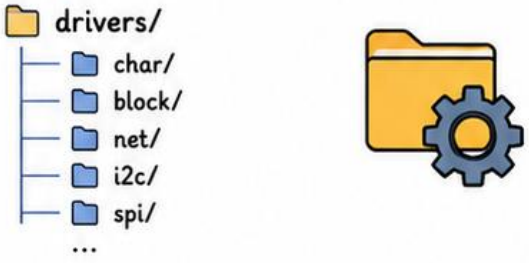
```
Module          Size        Used by
usb_core        245760      3
ext4            147456      1
i2c_dev         16384       0
                ⋮
```

6 Q6. Where are Device Drivers Located?

In the Linux Kernel source:

```
/drivers/
```

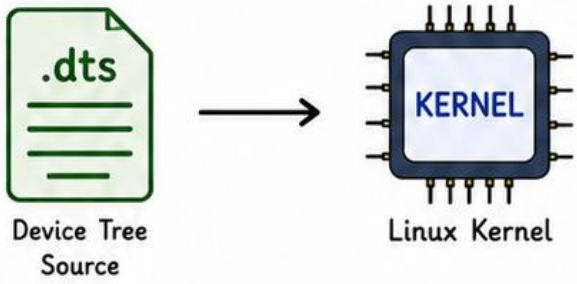
Example:



DEVICE TREE (DT)

1 Q1. What is Device Tree?

Answer: Device Tree (DT) is a data structure used in Embedded Linux to describe hardware and its configuration to the kernel.



2 Q2. Why is Device Tree used?

- It allows kernel to know about hardware at boot time.
- It makes the kernel generic and portable.
- It separates hardware description from kernel code.
- It is easy to modify without changing kernel source.

```
  Device Tree File
  (example.dts)
/ {
   compatible = "myboard";
   model = "Demo Board";
   soc  {
     uart0: serial@4000 {
        compatible =  "arm,pl011";
        reg = <0x4000 0x1000>;
        interrupts = <33>;
     };
   };
};
```

3 Q3. What is the extension of Device Tree source file?

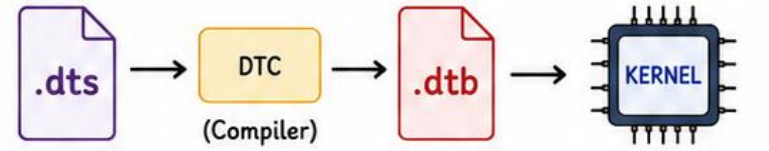
Answer: .dts (Device Tree Source)



Compiled to .dtb (Device Tree Blob) using Device Tree Compiler (DTC)

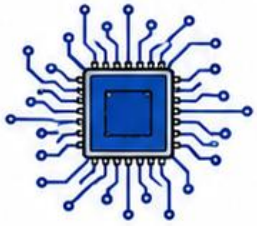
4 Q4. What is .dtb file?

Answer: .dtb is the compiled binary format of the .dts file which is loaded by the bootloader and passed to the kernel.



5 Q5. Where is Device Tree used?

- ARM, RISC-V and PowerPC architectures
- Used by Bootloader (U-Boot)
- Used by Linux Kernel during boot
- Common in SoC based Embedded Systems



6 Q6. How to view Device Tree in Linux?

```
dtc -I dtb -O dts -o output.dts /sys/firmware/fdt
```

This command converts .dtb to human readable .dts file.

```
$ dtc -I dtb -O dts -o output.dts /sys/firmware/fdt
$ cat output.dts | less
```

BUSYBOX

1 Q1. What is BusyBox?

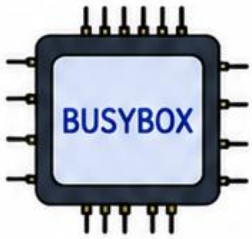
BusyBox is a single binary that provides many common Unix utilities in embedded Linux.



One Binary
=
Many Commands

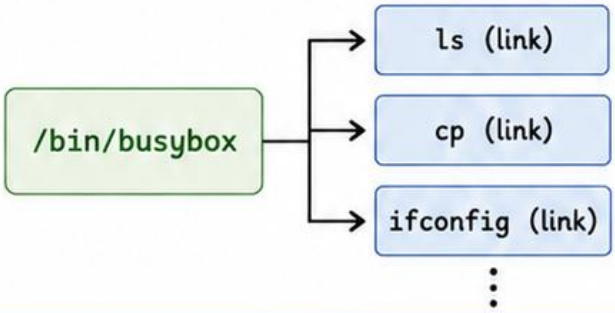
2 Q2. Why is BusyBox used in Embedded Linux?

- It reduces the size of the root file system.
- It saves memory and storage.
- It provides essential commands for system operation.
- It is ideal for resource-constrained devices.



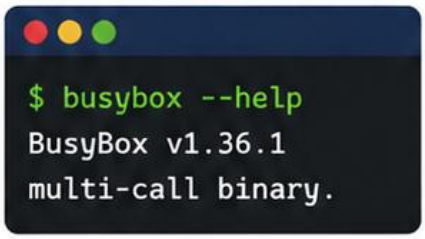
3 Q3. How does BusyBox work?

BusyBox creates symbolic links for each command to itself.
When a command is executed, BusyBox behaves accordingly.



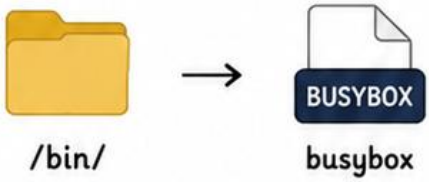
4 Q4. How to check BusyBox version?

```
$ busybox --help
BusyBox v1.36.1 (2024-01-01 12:00:00 UTC)
multi-call binary.
```



5 Q5. Where is BusyBox usually located?

```
/bin/busybox
```



6 Q6. Examples of commands provided by BusyBox

ls	cp	mv	rm	mkdir	rmdir	cat	echo
ps	top	ping	ifconfig	route	sh	vi	...