How to Start with Embedded Linux: 4 Steps

TechRounder PDF Edition

Live article: https://www.techrounder.com/development/how-to-start-with-embedded-linux-4-steps/

By Vipin PG | Published July 11, 2023 | Updated March 8, 2026 | Format: Guide | 4 min read

Quick answer

To start with embedded Linux, build your system using the Yocto project for customizable Linux distributions, then boot it through stages including bootloader launch, kernel loading, and root filesystem mounting.

If you want to build an embedded system, but don't know what technology to choose, get our tips on how to start with embedded Linux. Embedded engineers choose this OS for its versatility, customization options, and simplified development process. You can benefit from numerous embedded Linux services, including system design, GUI development, OS migration, and product development from scratch.

You can learn about embedded Linux and how to build, boot, deploy, and test an embedded system using this technology in this article.

Build a complete embedded system

Developing an embedded system from scratch is a demanding task. This is why you can use embedded Linux projects that provide various tools for engineers to facilitate the development process.

The Yocto project is an open-source project offering ready-made templates (configuration values, recipes, etc.) for Linux engineers. Yocto enables customization since embedded engineers can modify and tailor existing Linux distributions to the project's requirements. Numerous engineers around the globe have hands-on experience with the Yocto project and can help you build a functioning embedded system just the way you want it.

Such flexibility allows developers to undertake enterprise projects and easily integrate custom features. Usually, embedded Linux development engineers alter Linux distributions regarding clients' needs, minimizing customization challenges.

With the Yocto project, embedded engineers can add unique features to an embedded system without compromising performance. Besides, these features can be implemented during different project phases since they don't overlap with other system parts and don't cause system errors. You can prioritize releases and expand your system's functionality in the preferred order.

The Yocto project ensures strong device support by hardware manufacturers. If you choose to develop your system with Yocto, you will have broader component choices than with any other embedded Linux project aimed at facilitating engineering flow.

Boot your Linux-embedded system

Booting an embedded system is responsible for an embedded engineer's work. Developers check whether the client's project works the way it is supposed to work.

Bootloader launch At first, engineers launch a bootloader, which runs the operating system. Moreover, the bootloader allows you to select how to boot your system. This way, developers can test and debug a system regarding boot failure. Bootloader also ensures recovery options through a failsafe mode, which allows for minimizing the issues during system downtime. System initialization Next, the bootloader starts system initialization by preparing hardware components for proper functioning. Initialization is a process of early system configuration to ensure it works correctly. During this phase, the bootloader performs essential tasks to set up the hardware, load the operating system kernel, and configure various system parameters. Kernel loading After the bootloader has prepared the system for work, it starts kernel loading by loading it into memory. The kernel manages hardware resources, provides system services, and runs user programs. The bootloader transfers control to the loaded kernel, transferring necessary information. Initialization Once the kernel is loaded, it starts further initializing the remaining hardware. It also sets up device drivers and prepares the system for operation. The kernel detects and configures peripherals like network interfaces, storage devices, and input/output devices. Root filesystem mounting The root filesystem provides access to the various programs, libraries, and configuration files. The kernel mounts the root filesystem, which contains the necessary files and directories for the operating system to function. By mounting a filesystem, the kernel attaches it to a specific directory (the mount point).

This way, the files and directories within that filesystem become accessible. The mount point is an existing directory within the directory hierarchy that serves as the entry point for accessing the files and directories within the mounted filesystem.

Init process After the mounting, the kernel starts the init process, which launches other processes, starting system services and setting up the user environment. The init process helps to monitor and manage system services throughout the operating system runtime. For instance, it can restart failed services, track the status of running services, and handle dependencies between different services.

As a rule, engineers follow these stages to make an embedded Linux system function. However, this process varies depending on the complexity of the system and its configurations.

Deploy software to your embedded device via OTA updates

Once you boot your embedded system, there is one more way to deploy software - OTA updates. This way, getting started with embedded Linux is easier since you don’t need to access a device physically to improve its performance and fix the bugs. OTA updates allow to expand the functionality of an embedded system and deploy new features remotely.

Test the system

With embedded Linux getting started, you must test your embedded system to check whether it functions as planned. Embedded engineers can choose between on-host and on-target testing. During on-target testing, they will analyze device performance by testing it directly. With on-host testing, developers test the system while a device isn’t on site.

On-host testing is applicable if you cannot send your hardware to your engineering team. On-target testing is more effective since engineers can check how the software operates on the device and which bugs to fix to make it work better. Besides, on-target testing can reveal if you need to replace any components to enhance an embedded system. However, it can be costly to provide a device to a team.

Developing an embedded system depends on project requirements, documentation, and hardware components. Your embedded engineering team will help you explore unique approaches to your embedded project and create a dream product.

References

1. lembergsolutions.com - embedded-linux-development - https://lembergsolutions.com/embedded-linux-development

2. techtarget.com - searchmobilecomputing / definition - https://www.techtarget.com/searchmobilecomputing/definition/OTA-update-over-the-air-update