FreeRTOS vs Linux for Embedded Systems

ByteSnap

Which operating system is best suited for your embedded systems project – FreeRTOS or Linux?

FreeRTOS and Embedded Linux are two popular options for operating system provision in embedded systems.

But which should you choose for your electronics product design?

In many cases, the decision about which system to use is preferential, as software engineers generally have more experience in one or the other.

In this article we will present an overview of each option, and look at their comparative strengths and weaknesses.

What is FreeRTOS?

Probably the most popular real-time operating system currently in use, FreeRTOS is currently freely distributed under the MIT license. The license provides a very small amount of restriction over its usage and distribution, and means that businesses do not have to expose proprietary IP when using the system.

FreeRTOS was developed over 15 years ago specifically for use in microcontrollers. Now supported by AWS, they claim to provide the 'best of both worlds' in that the FreeRTOS kernel is completely free but also fully supported.

FreeRTOS is generally known as the de facto RTOS and is supported by most major semiconductor manufacturers, including NXP, Renesas and ST.

```
/*!
 * @brief Application entry point.
 */
int main(void)
{
    /* Init board hardware. */
    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_BootClockRUN();
    NVIC_SetPriority(LPUART1_IRQn, 5);
    if (xTaskCreate(uart_task, "Uart_task", configMINIMAL_STACK_SIZE + 10, NULL, uart_task_PRIORITY, NULL) != pdPASS)
    {
        PRINTF("Task creation failed!.\r\n");
        while (1)
            ;
    }
    vTaskStartScheduler();
    for (;;)
        ;
}
```

About Embedded Linux

Linux was first released in 1991, and is now the most used OS globally.

Embedded Linux is normally distributed under distributions such as Yocto Project and Debian. Both distributions are based on the Linux Kernel, but designed to be more light-weight and less feature heavy.

This enables engineers to choose the tools that are right for their project and allows great flexibility.

Linux is completely open-source, and is similar to Unix. Linux is very well supported across the board, by chip manufacturers, the many distributions, and by a global community of engineers and hobbyists.

For more information about some individual distributions, click here.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>
#include <strings.h>
#include <poll.h>

int main(){
        int fd = open("/dev/ttymxc2", O_RDWR|O_NOCTTY);
        if(fd < 0){
                perror("couldn't open serial port");
                exit(1);
        }
```

Real Time Operating Systems

To begin the comparison of these two operating systems we will first define real time OS', because although Linux has real time distributions, it is primarily used without this function.

Operating systems by and large can appear to run multiple programs simultaneously, which is known as 'multitasking'. However, as processor cores can only run a single thread or task at once, true multitasking is impossible.

This effect is therefore achieved using a scheduler – which is a program that decides which task to run next, and by rapidly switching between programs, creates the illusion of simultaneity. Different operating systems have different priorities, and these help the scheduler determine the order of tasks.

The priority of an RTOS is predictability. This is described as deterministic and requires tasks to be completed within a fixed timeframe or deadline. Due to this guarantee of completion, the operating systems behaviour can be anticipated. Embedded devices in particular have real-time requirements and this is partly why RTOS' are so popular in the sector.

Linux has a more performance optimized scheduler, which prioritises foreground tasks with the aim of improving overall device performance. As this is less deterministic than an RTOS, it is difficult to know when a task is going to be completed.

The RTOS classification can be further broken down in hard, firm and soft determinism. This refers to the consequences if a deadline is missed. The definitions of where the line is drawn between these grades is fairly arbitrary, but as a rule of thumb; if a deadline is missed in a hard RTOS, the system will breakdown or fail, whereas a soft RTOS will be degraded but will carry on functioning.

FreeRTOS uses a pure priority scheduling system and therefore within the limitations of the hardware, should be able perform as either a hard or soft RTOS.

Memory Management

There are two different types of processor commonly used in embedded systems: application processors and microprocessors.

The different between these is that application processors have a memory management unit. An MMU virtualizes and 'looks after' memory for you. As the memory is virtual, an MMU is able to prevent memory fragmentation and the subsequent slowing down of the system by mapping data across to other parts of the same program after memory has been freed.

Another advantage of an MMU is memory protection, which manages the memory access rights of programs, and helps with debugging.

Where do Linux and FreeRTOS come in? Embedded Linux requires an MMU to run, whereas FreeRTOS does not. Having the MMU decreases development effort and improves system support, however increases the memory footprint of the application.

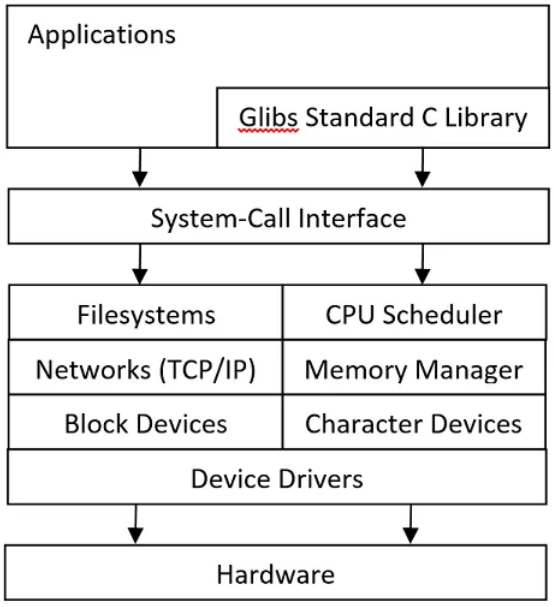
Memory

The key advantage currently of FreeRTOS is that it is lightweight. All the code is in one application layer and therefore less flash and RAM is required for a smooth experience.

Linux, by comparison, is large and clunky, with multiple code layers. The BSP in particular requires a lot of memory, and the OS can be slow to boot due to the complex file structure and amount of code.

Development Effort

FreeRTOS is more difficult to develop, due to the lack of the MMU. Whenever the device hits an error or snag it simply reboots, which makes it very difficult to identify bugs, and leads to a more frustrating development journey overall.



Flexibility

Linux is definitely the stronger contender here. When developing an embedded system, it is much simpler to find a driver for your hardware, and these are often provided by the manufacturer. There is more supported hardware for Linux than FreeRTOS currently, and therefore there is more choice and better opportunity for optimization without extending development time.

Furthermore, as Linux is a larger and more feature-rich OS there is a greater range of products that it is easier to develop for, including AI features and high resolution touch-screen displays.

Embedded Linux also has many programming language options, including Java, C#, C and C++.

FreeRTOS is only written in C and C++, which can be frustrating for some engineers, however most embedded applications are written in one of these two languages so this is usually not a problem.

Debugging

As stated above, the MMU makes debugging much easier for Linux, however debugging as a whole is a very similar experience for both Embedded Linux and FreeRTOS.

With the two operating systems the debugger connects differently; with Linux, the program is run through the debugger, and the IDE connects to it. With FreeRTOS, engineer's often

use JTag.

Unit Cost

FreeRTOS wins here – as it does not require an MMU to run, microcontrollers can be used, which are cheaper than application controllers. FreeRTOS simply has less code and therefore a smaller memory footprint, so less RAM and flash memory are also required.

Support

Linux currently is better supported. Drivers and Daemons for most hardware can be found online, so whatever the project, the engineers have a starting point. There is also a very active forum, where thousands of questions have been answered.

However, choosing FreeRTOS means gaining the benefits of a regular and periodic support updates from AWS, and the fact that currently it has reduced popularity actually could help with security, as it is a less popular target for hackers.

Also, while Linux is better supported at a whole, Amazon's ownership of FreeRTOS makes it perfectly placed for integration with Amazon products such as Alexa. AWS also has an advanced over-the-air update system that can be used with FreeRTOS, making it a good candidate for use in IoT systems.

FreeRTOS is also quickly catching Linux up with supported hardware and in the wider community, so this section may need altering in the next couple of years!

Porting

There are many reasons why a business may wish to port FreeRTOS over to Linux or vice versa. For Embedded Linux to FreeRTOS, this is usually to leverage the cheaper hardware to lower unit costs, especially for high-volume products.

The work-flow for porting Linux to FreeRTOS runs a little bit like this:

Hone down memory footprint

Remove or adapt LibC functions to work with FreeRTOS. FreeRTOS does not have the majority of LibC functions used to make Linux easier. In fact, leaving them in when porting can corrupt memory and cause the program to crash.

Modelling code – as the original application was written to run on Linux, it will be too different at this stage for a clean transfer. Therefore, code must be added to model the application as Linux within FreeRTOS to make it work.

Re-write network related code.

Reasons for porting to Linux from FreeRTOS may be to take advantage of the enhanced feature set and support to run display or more resource hungry hardware.

Porting to Linux is simpler, as hardware access can be done with user-mode Linux drivers. In a way, the application will do this itself. However, the same network recode will be required.

Summary

In conclusion, both operating systems clearly have key advantages over the other that make them better suited for different projects. They both allow full control of the hardware, and have similar development time when starting with equal experience.

However, most software engineers are currently more experienced with Linux. In addition, Linux comes with the extra support and flexibility of distribution choice and supported hardware.

Overall, we would recommend Embedded Linux as a starting point.

Nevertheless, FreeRTOS remains a valuable option for many applications – particularly as it continues to gain more support and catches up with Embedded Linux on functionality.

Need FreeRTOS or embedded Linux development support for your product design?

You're in safe hands with ByteSnap. Contact our experienced technical team with your requirements.