Linux OS: Embedded Systems

Author: Bronson Edralin

Date: 10/27/14

Abstract: Linux has made a huge impact in the computer industry and has paved the way for many applications. Linux paved the way for more embedded systems. With it being open-sourced, the community was able to develop real-time operating systems, which opened up so many possibilities. Many other operating systems for embedded systems were created and the open-source community just grew even more.

1 Introduction

Linux is an operating system that has made a huge impact in the computer industry. Many variations of Linux may be seen in a variety of applications such as servers, embedded systems, smart phones and tablets. The purpose of this writing assignment is to explore and research on how this revolutionary operating system played an effect on embedded systems. Topics such as history of embedded systems, real time operating systems and other embedded operating systems will be explored. Examples of real time operating systems include FreeRTOS, RTLinux and VxWorks. Embedded operating systems such as TinyOS and Raspberry Pi (RPi) will also be explored [2].

2 Embedded Systems and its History

An embedded system is a special-purpose computer system that is designed to perform very small sets of designated activities. Embedded systems were used as early as the late 1960s where they were used to control electromechanical telephone switches [6]. The first recognized embedded system was the Apollo guidance computer developed by Charles Draper and his team [6]. Later, embedded systems found their way into military, medical sciences, aerospace and automobile applications. Today, they are widely used in network equipment, consumer equipment, household appliances and mission-critical systems such as satellites.

There are key factors that differentiate an embedded system from a desktop computer. Embedded systems are usually cost sensitive and most have real-time constraints. Multitudes of CPU architectures such as ARM, MIPS, and PowerPC are used in embedded systems where they employ application-specific processors [6].

For example, the processor in your digital camera is specially tailored for image capturing and rendering. Embedded systems require very few resources in terms of RAM, ROM or other I/O devices as compared to a desktop computer [6]. Power management is also very important for embedded systems. The development and debugging environment in an embedded system is very different from a desktop computer. Embedded systems generally use a circuit for debugging purposes. Taking into account a specific application or set of applications, an embedded system is designed from the hardware and software perspective.

3 Real Time Operating Systems (RTOS)

A real-time operating system (RTOS) is an operating system (OS) intended for real-time applications. It should be able to process data as it comes in real-time without buffering delays. The operating system controls the allocation of the computer system's resources such as processor cycles, memory, communications and security [3]. The operating system must satisfy the competing demands for resources required by many different functions that the computer supposed to perform.

RTOS must be highly reliable. Microsoft Windows, MacOS, Unix, and Linux may often crash or freeze. Therefore, traditional operating systems are unsuitable for use in real-time systems [3]. RTOS must be able to work with minimal resources but traditional operating systems require millions of bytes of memory and expensive high-speed microprocessors to perform even the simplest tasks [3]. For example, a simple inkjet printer would be really expensive if it had a traditional operating system on every printer [3]. With the open-source Linux community, real-time operating systems were created.

FreeRTOS is an open source RTOS for embedded systems. It supports many different architectures and compiler toolchains and is designed to be small, simple and easy to use [1]. It is under active development since Richard Barry started work on it in 2002. FreeRTOS's main job is to run tasks where most of the code involves prioritizing, scheduling and running user-defined tasks [1]. Unlike all operating systems, FreeRTOS is a real time operating system, which runs on embedded systems.

RTLinux is a hard real-time RTOS microkernel that runs the entire Linux operating system as a fully preemptive process. Being a hard real-time system means that failure to meet deadlines will be fatal to the system. It was developed by Victor Yodaiken, Michael Barabanov, Cort Dougan and others at the New Mexico Institute of Mining and Technology and later developed as a commercial product at FSMLabs [7].

VxWorks is a RTOS developed as proprietary software by Wind River of Alameda, California, US. It was first released in 1987 for use in embedded systems requiring real-time and deterministic performance by industries such as aerospace and defense, medical devices, industrial equipment robotics, energy, transportation, network infrastructure, automotive and consumer electronics [8].

In a real-time kernel version of Linux, the scheduler has three scheduling policies: Normal, FIFO and Round Robin [4]. The Normal scheduling policy tries to implement fairness with no priority by using the completely fair scheduler algorithm. Completely fair scheduler handles CPU resource allocation for executing processes and aims to maximize overall CPU utilization while also maximizing interactive performance. FIFO (First in First Out) and Round Robin both implement the priority concept, but Round Robin uses time expiration periods for every process.

4 Other Embedded Operating Systems

The TinyOS is a flexible, application-specific operating system for sensor networks, which form a core component of ambient intelligence systems [5]. Sensor networks consist of thousands of tiny, low-power nodes, each of which executes concurrent, reactive programs that must operate with severe memory and power constraints [5]. The sensor network challenges of limited resources, event-centric concurrent applications and low-power operation drove the design of the TinyOS [5]. The operating system was able to meet those challenges and over a hundred groups around the world are using it today. TinyOS is able to become the platform of choice for sensor network research with it having low memory and low power requirements.

The Raspberry Pi (RPi) is a low-powered inexpensive, credit card sized computer that can use a computer monitor, a keyboard and mouse. It enables people of all ages to explore computing and allows the user do everything that you may expect a desktop computer to do. It supports a vast array of lightweight Linux operating systems, but Raspbian (Debian Wheezy) is the most popular operating system used on the Rasberry Pi. It definitely has the bare minimum amount of software packages, but you can always install more if needed. It has the ability to interact with the outside world and has been used in a wide variety of projects. The community for the Raspberry Pi is big and still growing with a vast array of different projects. The applications using Raspberry Pi are endless. I used the Raspberry Pi to control instruments and automate tests from here in University of Hawaii to test equipment in Indiana University.

5 Conclusion

As you can see, Linux has made a huge impact on the computer industry with many different applications and help pave the growth for embedded systems. Real-time operating systems (RTOS) such as FreeRTOS, RTLinux, and VxWorks made it possible for real-time embedded systems. The open-source community that Linux created made all of this possible. In a real-time kernel version of Linux, the scheduler used three scheduling policies: Normal, FIFO and Round Robin [7]. More embedded operating systems were created such as TinyOS. This helped push the sensor network research forward. Raspberry Pi (RPi) was also created and the applications are endless for this low-powered inexpensive, credit card sized computer. The RPi may be used for instrument control or even playing computer games. The Linux open-source community will continue to grow and continue to help evolve the embedded systems of today.

6 References

[1] C. Svec, (2014), *FreeRTOS* [Online]. Available:

http://www.aosabook.org/en/freertos.html.

[2] "EE468-Writing-Embedded" Writing Assignment 2 attachment for EE468,

University of Hawaii, Fall 2014.

[3] Green Hills. (2014). *The Leader in Real Time Operating Systems* [Online].

Available: http://www.ghs.com/RTOSLeader.html.

[4] L. Thang. (2012, April). *Comparing real-time scheduling on the Linux kernel and an RTOS* [Online]. Available: http://www.embedded.com/design/operating-systems/4371651/Comparing-the-real-time-scheduling-policies-of-the-Linux-kernel-and-an-RTOS-.

[5] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *TinyOS: An Operating System for Sensor Networks* [Online]. Available:

http://www.cs.berkeley.edu/~culler/papers/ai-tinyos.pdf

[6] P. Raghavan, A. lad, S. Ieelakandan. *Embedded Linux System Design and Development*. 1st ed. Florida: Taylor and Francis Group, 2006.

[7] Wikipedia, (2014), *RTLinux* [Online]. Available:

http://en.wikipedia.org/wiki/RTLinux.

[8] Wind River. (2014). *Backgrounder: Powering Innovation Since 1981* [Online].

Available: http://www.windriver.com/announces/wr30/documents/wind-river_backgrounder_0411.pdf.