The History and Prospect of Embedded Operating System

Wei-Min FANG\textsuperscript{1,a} and Xiang-Ning CHEN\textsuperscript{2,b,*}

\textsuperscript{1}School of Electronic Science and Engineering, Nanjing University, Nanjing, China
\textsuperscript{2}School of Electronic Science and Engineering, Nanjing University, Nanjing, China
\textsuperscript{a}email:fwwmmm@163.com, \textsuperscript{b}email: shining@nju.edu.cn

*Xiang-Ning CHEN

Keywords: Embedded Operating System; Historical Evolution; Internet of Things.

Abstract. With the advent of the era of intelligence, the concept that Internet of Things is proposed. As the cornerstone of the Internet of Things, embedded operating system is bound to become a major focus of the IT community. Through literature research, in chronological order, reviewed the history of the development and evolution of embedded operating system, listed the characteristics of the mainstream embedded operating systems at different times, and looked forward to the new opportunities of embedded operating system in the era of Internet of Things. In the end, discussed the development of embedded operating system.

The Development of Operating System

What is the Operating System

Modern computer system is a complex system which integrates system software, support software and user software. However, when the computer was first invented, there is no corresponding software system running on the hardware system. It only provides user with the corresponding hardware interface. User can effectively run the physical machine only after fully understanding the implementation details of the physical interface and be familiar with machine instruction programming. This makes it too difficult to use computer and results in low efficiency.

An operating system is a most basic system software which runs directly in bare machine. It is used to manage and control computer hardware and software resources. It is a computer program that is used to load the application and control its execution. It is the interface between software and hardware and the bridge between user programs and computer hardware. With the help of the operating system, we can consider a computer as a set of resources that can accomplish the functions of data movement, storage and processing. The responsibility of the operating system is to manage the hardware resources efficiently and effectively, and make sure any other software is running under the control of the operating system.

Phylogeny of Operating System

Computer System without Operating System

In 1945, the world's first generation computer was born. This computer is made up of thousands of tubes. At this time, there is no concept of operating system. Computer operation mode is that the programmers directly use computer hardware system in manual operation mode. The programmer prepares the tape which have perforated into the tape reader, then starts the tape reader and makes the procedures and data input computer, then the computer performs calculations. The results will be recorded on punched tape by tape puncher while the program is running automatically. After the programmer takes away the results of the calculation, the next user can then continue to use the computer.
There are two main disadvantages of this manual operation mode, one is that the user owns the entire machine, the other is that the CPU waits for manual operation, wasting the computing resources. Therefore the utilization of computer resources is very low in this way. In order to solve the contradiction between man and machine, and the speed mismatch between CPU and I/O devices, an offline input/output technique appeared in the end of 1950s.

Simple Batch Processing System

One of the earliest OS is simple batch processing system. Strictly speaking, it is not the OS what people now understand and it can only be regarded as the predecessor of OS. Because of the fact that early computer system is very expensive. In order to make full use of computing resources, the system should be ran all the time as far as possible to reduce idle time. For this reason, people usually mount a program called the monitor program onto the system firstly, then a batch of jobs are input to the tape in offline mode. Under the supervision and control of this monitor program, these jobs can be processed effectively and continuously.

Multiprogrammed Batch Processing System

In a single batch processing system, there is only one job in memory, once the IO is blocked, the processor enters waiting state, and a large number of computing resources have lain idle. With the advent of multiprogramming in the mid-1960s, multiprogrammed batch processing system gradually formed. In multiprogrammed batch processing system, there is a job queue called reserve queue in the computer disk to store the user submitted jobs. In order to share the resources of CPU and system, job scheduling procedures may select several jobs into memory from the reserve queue according to some special scheduling algorithm. This greatly improves the processor utilization and avoids the waste of computing resources [1].

Time-Sharing System

In order to realize the function that multiprogrammed batch processing system cannot provide, such as human-computer interaction or shared host, MIT developed the first time-sharing operating system. After that, MIT and Bell laboratory, GE Corp developed a multi-user multitasking operating system. Because multiple users share processor time, it was referred to as time-sharing system.

The Development of Embedded Operating System

What is Embedded System

There are two kinds of definitions with embedded system. One is that the embedded system is a special computer system which, based on computer technology, is applied as hardware centric. Its software can be cut and has strict requirements on function, reliability, and volume, cost and power consumption. The other is that the embedded system is a combination of software and hardware, and may also have a mechanical device or other components used to achieve a specific function.

We believe that the embedded system is a kind of intelligent processing system which is integrated into the hardware of the application equipment and is used to implement a specific application function. The processing core has a certain computing power, and can be used to realize the real-time monitoring of the external equipment state and receive external information and real-time control of other external devices or devices through the built-in interface. The main task of embedded system is to enhance the ability of the device to adapt to the change of the application environment, rather than the traditional computer system to undertake the heavy computing tasks.
What is Embedded Operating System

Each embedded system has at least one embedded microprocessor (or MCU and DSP). The software running on these embedded microprocessors is called embedded software. These early software is not complicated. Engineers can complete some specific tasks through the bare metal program. However, with the development of embedded microprocessor and micro controller from 8 to 16 and 32 bits, the embedded system becomes more and more complex. Therefore, it is necessary to have an operating system to manage the embedded software, and provide the application programming interface for the development of the functional module. All operating systems that can be used for embedded systems can be called embedded operating systems. Since it is an operating system, it must have the function of the operating system, including the task-loading, communication, scheduling, memory management, time management and other kernel functions such as the file system, networking support, device drivers and other services functions.

The Development of Embedded Operating System

The Competition of RTOS Era

In the late 1970s, the concept of real-time multitasking kernel was proposed. In 1980s, while the application of embedded system becomes more and more complex, the embedded operating system with real-time multitask kernel cannot meet the requirement of embedded development. The real time multitasking kernel begins to develop into a complete real-time multitasking operating system (RTOS) that includes a network, file, development and debugging environment.

Nowadays, RTOS has formed an industry in the world. The world's first commercial embedded real-time kernel (VRTX32) was developed by Ready System in 1981. Then in 1993, Ready System and the famous Silicon Valley embedded software company Microtec Research Merger developed two new RTOS kernel VRTX32 and VRTXsa on the basis of VRTXmc. At the same time, VRTX integrated development environment (Spectra) appears, which greatly improves the development of embedded software.

At almost the same period, in 1989, in order to design an ignition system based on microcontroller, Mr. Jean Labrosse began to develop his own microcontroller core, and released the µC/OS in a year. Then µC/OS-II is developed on the basis of µC/OS.

µC/OS-II is a fully functional multitasking embedded real-time operating system. In the aspect of task scheduling, it uses static priority preemptive scheduling, which cannot have the same priority at any time. This simple scheduling method can meet the high real-time requirements and can support up to 64 tasks running at the same time. In memory scheduling, µC/OS-II uses improved partition management. Large contiguous memory is divided into a plurality of regions, each region also contains a number of memory blocks with the same size. The size of memory block in different memory area is different, so that applications with different size can be combined with different memory blocks according to the needs. It is important to note that all memory blocks should be released when the program is running completely. Using this mechanism, µC/OS-II effectively avoids the generation of memory fragmentation [2].

The other side of the banner in the field of RTOS is the Wind River VxWorks. VxWorks is originally a complete embedded operating system and development environment based on VRTX kernel. On this basis, the Wind River has developed the industry's first RTOS kernel, by using C programming language, and in 1989 officially released their own embedded operating system VxWorks. Vxworks uses micro kernel structure, which is characterized by only a small core of the operating system to achieve the most basic functions. Other functions are in the user state, outside the core as far as possible. The micro kernel structure can make the structure of the operating system clearer, which is helpful to improve the stability and portability of the operating system [3].

Microsoft, which dominates the desktop operating system, also released its own embedded
The embedded operating system is WinCE, which is originally based on Windows 95 in the 1.0 version, and subsequently 2.0 version, 3.0 version, until up to 6.0 version, supporting x86, ARM, SH4, MIPS and other processor architecture.

WinCE is a preemptive multithreaded operating system. Through the thread, WinCE is better to achieve the embedded operating system task scheduling, inter-task communication, mutual exclusion and other functions. In order to solve the problem of priority inversion in preemptive real-time scheduling in WinCE, priority inversion mechanism is realized [4].

**Desktop Linux and Embedded Linux**

Linus Torvalds developed Linux open operating system in 1991. Since Linux is a Unix class operating system, it is powerful and compatible with the POSIX standard, and is highly portable. So it has been a great success in the field of servers and desktops.

As a general-purpose operating system, throughput must be considered in design. Especially in the single processor mode, preemptive scheduling algorithm involves a large number of critical area operation and the extra overhead has a serious impact on system throughput. As a result, general Linux does not support preemptive scheduling.

However, for embedded Linux in real-time environment, it is necessary to take high priority real-time tasks to overtake other tasks with lower priority according to task priority, so as to ensure the real-time performance. Embedded Linux adopts two schemes to solve this problem.

The first method is the preemptive point method. In order to fulfill the need of real-time tasks, embedded Linux need to set up a series of preemption points in the kernel running path so that the longest delay of the highest priority real-time task is the longest path between two preemption points. This method is implemented in Red Hat's Ingo Molnar kernel patch.

The second method is to transform the desktop Linux kernel into a preemptive kernel. When a task with high priority in the task queue needs to be run, and current task is not in the situation that is walking in the critical area, high priority task can directly seize the control of current task. This method is implemented in Monta Vista's Hard Hat Linux.

In memory management, Linux supports virtual memory. When the application is running, you can implement on-demand paging so that the computer can use the memory resources efficiently. However, the virtual memory mechanism needs the support of the memory management unit (MMU). Most embedded operating system is running on the MCU with limited on-chip resources. These MCU usually do not contain MMU. On the other hand, the virtual memory mechanism will cause some uncertain IO blocking time, which will reduce the real-time performance of the operating system. Therefore, the general embedded Linux does not support virtual memory. Lineo provided by the Clinux, support for memory management without MMU microprocessor [5].

In addition to the implementation of the embedded Linux operating system, IBM developed the next generation of Java integrated development environment (IDE) Eclipse in 1999 and it was contributed to the open source community in 2001. In fact, Eclipse is far more than the use of Java language. This relatively mature and developed application integration system, also makes it easier to promote embedded Linux. It has become one of the main tools of embedded development.

**The Operating System Dream of ARM**

When it comes to embedded operating system, we have to pay attention to the hardware system. Although the major semiconductor companies produce a wide variety of SoC, the famous SoC such as ST STM32, Samsung S3C2440, NXP semiconductors LPC800, are all the products based on ARM architecture. Today, the world's major semiconductor manufacturers have produced more than 60 billion ARM core chips.

October 2014, ARM announced the launch of its Internet of Things operating system mbed OS for low power embedded networking devices. ARM said that in order to ensure that the operating
system will not be fragmented, mbed OS will be part of open source. Mbed OS requires only 256KB ram, available for free use for all Cortex-M devices.

The Arrival of a New Era of Embedded

From 1981 the world's first commercial embedded real-time kernel appeared to now, more than 30 years has past. The development of embedded application has gradually gone from bare SCM era into the embedded system integrated circuit, computer, communications, electronics and other disciplines blend era. Today, with the development of the Internet, the concept of Internet of Things has put forward one hundred billion connections. Let us surf in the tide of the Internet of things era.

Nowadays thousands of embedded devices are used in the system of Internet of Things and hundreds of sensors are used in a car. Compared with the traditional embedded system, the Internet of Things system requires lower power consumption, safe and reliable, and has the ability of ad hoc network. The communication section needs to meet the conversion between various communication protocols and the application layer must have the ability of cloud computing. This puts forward higher requirements for the design of the operating system. To answer this, a software system for the Internet of Things which called Internet of Things operating system arises at the historic moment.

The origin of the Internet of Things OS from the wireless sensor network is the two open source OS named Tiny OS and Contiki. Tiny OS was an open source project launched in 2000 by University of California at Berkeley and Intel. Everyone can view and modify the Tiny OS source code, participate in the development of Tiny OS and its supporting software, and apply it to different areas of business and industry. Under the collaboration of many participants, Tiny OS released the V2.1.2 version in 2012, and it was posted to the open source project hosting site Github for global players to download and use. Contiki is an open source operating system developed by Thingsquare founder Adam Dr. Dunkels in the development of uIP/LWIP protocol. For this reason, Contiki is more focused on the network, which is the original intention of its design [6].

The purpose of the design of Tiny OS is to make it an operating system which belongs to the embedded wireless sensor network. The software engineering of Tiny OS is constructed by components, and it is also a set of components. At last, Berkeley launched a program called NesC programming language to achieve the development of Tiny OS and Tiny OS applications. A composite component formed by assembling a plurality of components according to a certain level of relation is a complete Tiny OS program. In the aspect of task scheduling, Tiny OS uses the two level scheduling of tasks and events. In the application where real-time requirements is not high and the task is used for transaction processing, the task is not mutually exclusive and is run as a single thread, the scheduling algorithm is a default FIFO scheduling with each task is assigned a separate task stack. In the application with real-time requirements, the event is used for transaction processing, and the event is equal with each other. It can preempt tasks or low priority events [7].

Contiki is a multitasking open source operating system and highly portable that supports the network. It consists of a multitasking kernel, an assembly, a simplified TCP/IP stack, and a low-power wireless communication stack. Contiki was developed by C language, only a few KB of memory can also load the entire system. Its original intention is to solve the problem of hardware memory limitation in wireless sensor networks system. Contiki also includes an optional GUI subsystem that provides graphical support for local terminals, Networked Virtual displays based on VNC, or Telnet. The kernel of Contiki is event driven, which can be dynamically loaded at runtime. The main process model of Contiki use single thread, the process model uses only one stack, so that Contiki can save the memory on this not very rich embedded device [8].

In the field of embedded development, although the two operating systems are both wireless
sensor network based on the networking operating system, but Contiki has many advantages. Contiki uses a common C compiler, such as GCC, IAR, etc. It supports 8 bit, 16 bit, almost all types of processors in the 32 bit and has become the first choice of the Internet of Things operating system.

In the field of Internet of Things operating system, there is a very famous operating system, FreeRTOS. It is a free open source real-time operating system with a kernel of embedded operating system. It has the characteristic of open source, high portability, scalable, and flexible scheduling strategy. It can be easily transplanted and run on a variety of SCM. FreeRTOS has the basic function of the operating system kernel, including task scheduling, memory management, time management, semaphores, message queues and other functions. As a lightweight operating system, it can satisfy the basic needs of small embedded system. The minimized FreeRTOS kernel contains only a total of less than 9000 lines of code, and a typical compiled binary code image is less than 10KB. In the aspect of task management, FreeRTOS uses a round robin algorithm based on the priority preemptive and it can effectively guarantee the concurrency of the whole system [9].

Today, the Internet of Things industry is still in the early stages of the development. The competition in the field of Internet of Things has just begun. The major companies have launched their own Internet of Things operating system for future. Microsoft launched Windows 10 IoT Core, the system takes 256KB RAM, 2GB Flash. It currently supports Intel Edison and two high-end processors with ARM architecture. Micrium released the Spectrum Internet of Things OS in October 2014. ARM also released mbed OS which attempted to replicate the success of Google Android in the mobile Internet field. Google also released an IoT underlying operating system Brillo in 2014 Google I/O conference and launched a special programming language Weave at the same time. Among them, Brillo is a lower level system than Android, which is based on Android. It supports a variety of protocols in Android including WiFi and Bluetooth low-power and can be run on 32MB or 64MB memory devices. HUAWEI launched a plan called agile network 3.0. As a part of it, HUAWEI launched its lightweight networking operating system Lite OS in the 2015 HUAWEI network conference. The system is in the 10KB class, as an open source information processing operating system suitable for a variety of small devices. This is also a key step in the construction of HUAWEI lightweight networking system.

**Concluding Remarks**

Throughout the 30 years development of the embedded operating system, there are hundreds of types of operating systems in the most prosperous period and different operating system emerge in an endless stream. After 30 years of development, many operating system disappeared, and the number of embedded operating system down to dozens, although still no one embedded operating system dominates like Windows in desktop market.

There is no hardware and software system like Wintel Alliance established by Microsoft and Intel in the field of embedded software. In other words, there is not a complete ecosystem of hardware and software. ARM focused on the design of the general hardware architecture and ignored the direction of the development of the system. It is very difficult for the major semiconductor manufacturers and embedded operating system companies to cooperate closely to promote the establishment of ecological rules in embedded operating system because of their own interests. The research of embedded operating system is not only an opportunity but also a challenge to the development of domestic embedded operating system. We should gradually establish the ecosystem of embedded operating system on the basis of defining the hardware architecture and embedded operating system software function. Perhaps it is the future of the domestic embedded operating system, and even the entire operating system.
References
[1] William Stallings. Operating Systems: Internals and Design Principles [M], sixth ed, Prentice Hall, 2008.
[2] Tian Lingping. The Research on memory management in embedded operating system [J], Computer knowledge and technology, 2004(4):447-453.
[3] Hu Shuhui, Chen Jian. Analysis and comparison of several embedded real time operating systems [J], MCU and embedded system application, 2007(5):5-9.
[4] Jin Liang, Zhang Xuejie. Analysis of key technologies of 3 embedded operating system kernel [J], Journal of Yunnan University, 2006, 28(S2):1-4.
[5] Liu Wenfeng, Li Chengyuan. Research on embedded Linux operating system [J]. Journal of Zhejiang University, 2004, 38 (4): 447-452.
[6] He Xiaoqing. New development of the Internet of things operating system [J]. Electronic products world, 2015.9.008.
[7] Wu Xiaona, Wang Xing. Review of Tiny OS for Wireless Sensor Networks Operating System [J], Computer and modernization, 2011 (2): 103-106.
[8] He Xiaoqing. Internet of things operating system analysis [J]. Microcontroller and embedded system applications, 2015 (1): 5-8.
[9] Liu Bin, Wang Qi, The principle and implementation of embedded operating system FreeRTOS [J]. Microcontroller and embedded system applications, 2005 (7): 8-11.