Embedded Operating Systems: Windows CE, Embedded Linux, pSOS, uC/OS

Kwang Hyun Park* and Jae Wook Jeon**

School of Information and Communication Engineering, Sungkyunkwan University, 300 Chunchun-Dong, Jangan-Gu, Suwon, Kyungki-Do, 440-746 Korea

*(Tel: +82-31-290-7937; Fax: +82-31-290-7937; Email: shaia@ece.skku.ac.kr)

** (Tel: +82-31-290-7129; Fax: +82-31-290-7231; E-mail: jwjeon@yurim.skku.ac.kr)

Abstract: Except a desktop computer and workstation, an embedded system is a system containing microprocessors. While a desktop computer and a workstation are designed for a general purpose, an embedded system is designed for a dedicated purpose. Thus, an embedded system must meet some constraints such as low power consumption, low cost, small size, real-time, or user-defined ones. A simple and low cost embedded system may be able to be designed without using embedded operating systems (OS). However, considered design time and effort, some embedded system had better be designed with using embedded OS. Under given constraints and purpose of some embedded systems, one embedded OS can save more time, cost, and effort in designing those embedded systems than others. This paper compares four embedded OSs, Windows CE, Embedded Linux, pSOS, and uC/OS. It analyzes several issues of embedded OS such as process scheduling, inter-process communication (IPC), memory management, and network support. Also, it describes the product of each embedded OS.

Keywords: embedded system, embedded operating system, Windows CE, Embedded Linux, pSOS, uC/OS

1. INTRODUCTION

The constraints and purpose of embedded systems are different from each other. As a result, appropriate software architecture must be selected for a given embedded system. There are two common types of software architectures to design embedded systems. One is to design them without using embedded OS. In this design the software simply has a loop. The loop calls subroutines and each subroutine manages some hardware or software. In designing a simple embedded system, this software architecture is better. The other is to design embedded systems with using embedded OS. In this design, an embedded OS can provide most basic functions. Since the characteristics of embedded OSs are quite different, one embedded OS may be better than others in designing an embedded system. Therefore, we had better understand the characteristics of each embedded OS. This paper compares and analyzes four embedded OSs, Windows CE, Embedded Linux, pSOS, and uC/OS.

1.1 Windows CE

In November 1996, Windows CE was first introduced in the handheld PC. After Microsoft (MS) released the windows CE OEM Adaptation Kit in March 1997, it became a highly configurable OS. Windows CE includes a low overhead device driver model, a built-in power management, and a subset of the Win32 API that can address most commonly needed services. Windows CE has been applied to many consumer electronics and many industrial systems [1].

1.2 Embedded Linux

Nowadays, Embedded Linux is popular in embedded systems as Linux is popular in desktop computers. Embedded Linux is made from Linux to meet an embedded system OS. Since Linux kernel is open and free, Embedded Linux is very attractive to embedded system designers. Furthermore, since it is modular, it is easy to slim down the OS environment by removing OS utility programs, tools, and other system services not needed in an embedded environment.

Some modules can be added or removed from embedded Linux kernel at runtime. Also, Linux is a fully functional OS with network support for network and it is portable to a wide range of microprocessors [1].

1.3 pSOS

Software Components Group became a pioneer vendor in real-time embedded OS area when its pSOS (Plug-in Silicon Operating System) product was introduced for 6800-family embedded designs in 1982. Just as one hardware designer would select and use a standard chip-level component, one software designer could readily plug “silicon operating system” or “software component” into an embedded system. PSOS is small, fast, and can be written in the ROM. Also, pSOS consists of position-independent binary object code and thus its software component could be positioned anywhere in memory. Its system services were accessed via trap instructions according to a well-defined and well-documented interface. Just as in the case of a standard hardware component, pSOS’s performance was carefully benchmarked and its complete timing references were available so that users could assess its ability to meet their system requirements [2].

Today, pSOS offers a complete line of standard OS building blocks including pSOS+, a real-time OS kernel component for embedded designs, a sophisticated system and assembly-level debug monitor, and a plug-in file system manager as well as several other companion products [3].

1.4 uC/OS

Jean J. Labrosse designed uC/OS. uC/OS is a fully preemptive real-time kernel and it always runs the highest priority task that is ready. Since uC/OS can manage up to 64 tasks but its current version reserves 8 tasks for system use, we can use up to 56 tasks. Each task has a unique priority assigned to it and there are 64 priority levels. According to each embedded system, we can choose some services of uC/OS, which saves the memory amount in the embedded system [4].

Four embedded OSs are analyzed and compared in the following points of view: hardware resources in section 2, architecture in section 3, the process handling in section 4, memory management in section 5, and network support and application products in section 6. In section 7, the conclusion is described.

2. HARDWARE RESOURCES

Processor and memory amount required in each embedded system OS are different. Some embedded OS can support more processors than other. Also, the memory amount required in one embedded OS is larger than others. Embedded system designers have to consider necessary hardware resources when they select one particular embedded OS.

2.1 Processor

Windows CE requires processors to use a flat 32-bit address space, to support kernel-mode and user-mode operations, to support virtual memory using an MMU, and to be little-endian. uC/OS supports more various CPUs. Since Embedded Linux is open, it can support more processors than Windows CE. Because the integrated development environment (IDE) tool kit is necessary in porting pSOS like Windows CE, pSOS cannot support some processor when its IDE does not support that processor. Since uC/OS is open and is smaller than other embedded OS, it can support more wide range of processors including 8-bit ones than other embedded OS. On the other hand, designers must make their own development environment [1, 3, 4]. Table 1 shows processors supported by each embedded OS.

Table 1. Processors in each embedded OS.

OS	Processors
WinCE	-Intel x86-compatible family (With MMU) -AMD : Elan SC400, SC410, SC520, 486DX : Au 1000, 1100, 1500 -MIPS : 4KC, 5KC, 40KC core -NEC : VR 4xxx series -Toshiba : TX39xx series -Hitachi SH3 and SH4 -Motorola/IBM PowerPC - ARM, Strong ARM
Embedded Linux	- Intel x86-compatible family - MIPS family - ARM, Strong ARM, X-Scale - Motorola/IBM PowerPC family - Motorola 68K series - Hitachi SH3 and SH4 - Sun SPARC
pSOS	Motorola/IBM PowerPC family - Motorola/IBM PPC6xx, 7xx, 74xx - Motorola MPC8xx, 82xx - IBM PPC403 MIPS architecture family - MIPS 16, 32 and R500 ARM RISC Machines
uC/OS	- Intel x86-compatible family - MCS-251, 80196, 9096 - Analogue Devices AD21xx series - ARM, StrongARM - Hitachi 64180, H8/3xx, SH series - Mitsubishi M16 and M32 - Motorola PowerPC, 68K, 68HC11, 68HC16 - Philips XA - Siemens 80C166 and TriCore - Texas instruments TMS320 - Zilog Z-80 and Z-180

2.2 Memory

Memory is precious and expensive resource on embedded systems. Thus, every embedded OS tries to reduce its memory occupation size. Among four embedded OSs, uC/OS needs the smallest memory size because it has a very small kernel about 3k bytes. Windows CE, Embedded Linux, and pSOS require more memory. Windows CE needs 350k bytes for a minimal system with its kernel and some communication support. Embedded Linux needs 125-256k bytes for its kernel and over 100k bytes for other components. PSOS needs 60-100k bytes for a reasonable configured kernel. However, the system memory size for each embedded OS may be varied depending on application product.

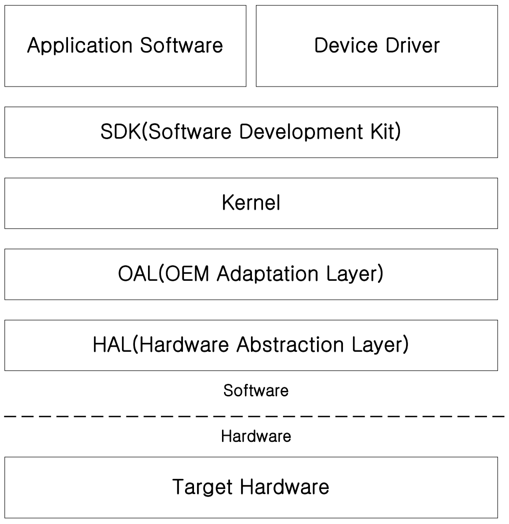
3. ARCHITECTURE

Windows CE and embedded Linux support multi processes, and pSOS and uC/OS support multi threads. Multi threads can access common memory area freely. Application programs shares the same address space with OS kernel and they can use kernel resources. Therefore, context-switching time is very fast. Since OS kernel and application programs are unified, all components are compiled together when OS Image builds. Multi thread model is suitable for a small embedded system. However, a trivial bug can stop the whole system. In pSOS and uC/OS, application programs cannot be added to an embedded system where porting is done. If we want to add an application program, we have to build all components that include kernel, existing application programs, and additional application programs. Then all components are compiled again. In the multi process model such as Windows CE and embedded Linux, application programs can be added to an embedded system where porting is done. Windows CE can use Active-Sync function for this purpose. In the multi process model, an application program is running in a different address space from the kernel address space. The process of an application program cannot use the kernel resources. Different from multi thread model, a bug in an application program does not stop the whole system. The multi process model is widely used for a complex embedded system that needs various application services [5].

While Windows CE and embedded Linux are used for soft real-time systems, pSOS and uC/OS can be used for hard real-time systems.

As in Fig. 1, the architecture of Windows CE is hierarchical [1, 6]. HAL(Hardware Abstraction Layer) and OAL(OEM Adaptation Layer) lie at the bottom. The OEM implements them when Windows CE is porting. Above them lie Graphics

Windowing and Events Subsystem (GWES), the kernel itself, and the communication stacks. Microsoft implements this layer. The Remote API capability is built on top of the communication functionality. On top of the kernel lies the database and file system. This is accessed by the RAPI calls and is made available to the applications via the Win32 interface. Each application program runs in its own address space and interacts with the rest of Windows CE via the Win32 system call interface.



Application Software
Device Driver
SDK(Software Development Kit)
Kernel
OAL(OEM Adaptation Layer)
HAL(Hardware Abstraction Layer)
Software
Hardware
Target Hardware

Fig. 1. Windows CE architecture.

Embedded Linux has a layering structure and consists of several modules as in Fig. 2. It takes the Linux kernel and extracts the necessary modules. Linux kernel consists of five subsystems: the process scheduler, the memory manager, the virtual file system, the network interface, and the inter-process communication. While each subsystem in embedded Linux is an independent component of the Linux kernel, there are some interactions among five subsystems [1, 7].

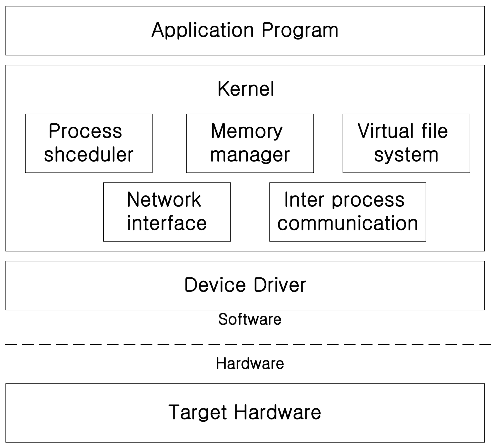


Fig. 2. Embedded Linux architecture

As in Fig. 3, PSOS has three main components in addition to its kernel: pROBE is debugging agent that supports remote debugging and target system debugging. pMONT takes charge in system usage monitoring and event logging. pREPC is a pre-defined function library that can be used in application [2,3].

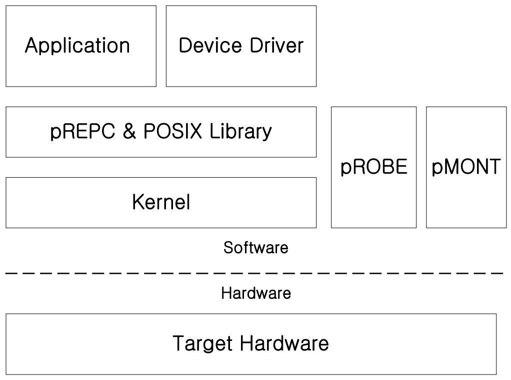


Fig. 3 pSOS architecture.

uC/OS is a very small micro-kernel real-time OS and it has only basic OS services such as task management service, time management service, inter task Communication (ITC) services, and memory management services. Fig. 4 shows the structure of uC/OS kernel [4].

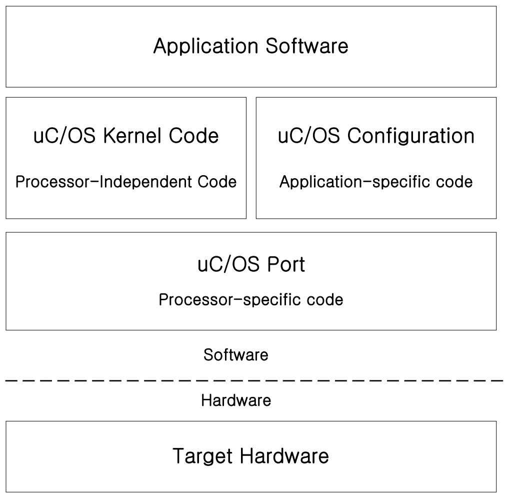


Fig. 4 uC/OS Kernel architecture.

4. PROCESS

4.1 Process or Task management

Windows CE is preemptive multitasking OS and it supports processes and threads. Preemption conforms to only thread priority. The same priority threads are scheduled according to round-robin method [1,6].

While embedded Linux is multi process model, Linux kernel is implemented with threads. Therefore, Linux scheduling is based on threads, not processes, in the kernel data structures. Linux has three classes of threads for scheduling: real-time FIFO threads, real-time round robin threads, and time-sharing threads. Real-time FIFO threads are the highest priority and these threads are not preemptive. Real-time round robin threads are also highest priority threads, but they can be preemptive. Timesharing threads have lower priority than the others. Each thread has scheduling priority. Embedded Linux is not fully real-time OS (RTOS). It is not fully preemptive kernel. By creating a RTOS kernel and running Linux as an unscheduled thread under that RTOS, embedded Linux can be made as RTOS [1,7,8].

pSOS is preemptive multi-tasking OS. Each task has priority and the highest priority task always runs first. That is, pSOS is priority-based preemptive OS. Total number of priorities is 256, 255 is highest priority task level and 0 is the lowest one. Kernel tasks have priorities from level 240 to level 255. ROOT task is 240 and IDLE task is 0. Round-robin scheduling method is applied to two tasks of same priority. The unique key of each task is Task ID that is Task Control Block's (TCB) pointer. User or programmer defines Task ID, and TCB is allotted to OS. Then, OS passes the pointer. Task's name is allowed to have four characters. In the point of user's view, task's name is unique ID about a task. Task consists of two steps: "Create" and "(Scheduling) Start". Created tasks must run after "Start". When task is created, four arguments can be passed to the task [2,3].

uC/OS is also preemptive multi-tasking OS. Each task has its unique priority and the highest priority task always runs first. UC/OS is priority-based preemptive OS likes pSOS. But, uC/OS does not support round-robin scheduling. uC/OS supports only priority-based scheduling method. Maximum tasks number is 64, but 0 to 3 and 60 to 63 tasks are reserved for OS, so there are 56 available tasks. Different from pSOS, user defines uC/OS's task ID and task does not have its name. Since the priority of each task is the same as its ID, the same task IDs cannot be used.

4.2 Inter Process or Task Communication (IPC, ITC)

Since Windows CE supports message passing and memory mapping between processes, it is possible to transfer data very fast between processes [1].

Embedded Linux provides several methods of IPC, which are the same mechanism as the original Linux IPC. That is, embedded Linux can use one of Signals, Wait queues, File locks, Pipes, Semaphores, Message queues, and Shared memory methods [1,8].

ITC (Inter Task Communication) of pSOS, is done by using Message Queue, Semaphore, Condition Variable, Mutex, Event, Asynchronous Signal Routine (ASR), and Task Control Block (TCB) [3].

uC/OS provides several mechanisms to protect shared data and it also provides several methods of IPC. One is Disabling and Enabling interrupts through the macros OS_ENTER_CRITICAL( ) and OS_EXIT_CRITICAL( ). Another is Locking and Unlocking uC/OS's scheduler with OSSchedLock( ), and OSSchedUnlock( ). The others are using Semaphore, Message Mailbox, and Message Queue [4,9].

5. MEMORY MANAGEMENT

Windows CE uses paged virtual memory that is the unit of protection and memory allocation. The use of virtual memory is closely related to multiprogramming. Each process operates in its own environment and address space. Since each process has its own page table, different page table is used on process switching. Due to memory protection among processes, Widows CE requires its processors to have MMU (memory management unit), which results in more complex processors [1,6].

Embedded Linux manages memory through Memory Manager Subsystem as in Fig. 2. It also uses virtual memory. One type of embedded Linux that does not require MMU is uClinux [1].

pSOS uses two conceptual ways to allot necessary memory to tasks. One is allotting various segments (or memory) that are not necessarily contiguous to task. It allots the same memory amount as each task asks. Since this is efficient memory utilization, it makes memory fragmentation problem and results in low system performance due to complicate inner mechanism. The other is allotting a multiple number of fixed amounts of memory. While this is not efficient memory utilization, it does not make memory fragmentation problem and results in high system performance. As in explained previously, pSOS does not support memory protection and every task share same memory address [3].

uC/OS provides an integral number of blocks. One type of block exists or several types of blocks may exist. All blocks of each type have the same amount of memory. If several types of blocks are used, it does not make memory fragmentation problems [4].

6. NETWORK SUPPORT AND PRODUCT

6.1 Network support

Windows CE supports SNMP (Simple Network Management Protocol), TCP/IP, NDIS (Network Driver Interface Specification), and WinInet that is used to access HTTP service and FTP service in an Internet application program. If there is no available DHCP (Dynamic Host Configuration Protocol) server, Windows CE supports Autonet function that allocates an IP address to one device. Windows CE also supports IrDA protocol for infrared communication, various kind of communication and network function [11].

Since embedded Linux supports most protocols used in Internet, it can support various functions needed in web server, E-mail server, DHCP server, DNS server, and FTP server. Embedded Linux supports network file system for Unix such as NFS or AFS, and it can also exchange data with Windows system because it supports Samba [12].

pSOS has a stand-alone TCP/IP protocol stack called pNA (pSOSystem Network Architecture). Many different network protocols through OpEN (Open Protocol Embedded Networking) can be used at the same time without mutual influence. pSOS can support a standard Remote Procedure Call through pRPC and it can perform a FTP, Telnet, NFS, Routing back through Net Utilities. Also, pSOS uses MIB-II and SNMP [3].

Since uC/OS does not have a network function, one designer must make the desired network function by software or hardware [4].

6.2 Application product

Fig. 5 shows one typical product of Windows CE. Windows CE is widely used in PDA, web-pad and handheld PC. The functions and features of Windows CE are suitable for mobile multimedia devices.

Fig. 5. COMPAQ: Pocket PC H3900.

Fig. 6 and 7 show one PDA and one smart phone, respectively, that use embedded Linux. Embedded Linux is also widely used in mobile multimedia devices.

Fig. 6. SHARP: Zaurus.

Fig. 7. Motorola: A760.

Fig. 8 and 9 show pSOS products, Bang & Olufsen company's Home-theater system and INTERFACE company's programmable serial communication card, respectively.

Fig. 8. BeoVision Avant TV.

Fig. 9. IC-SER-PCIb.

Fig. 10 and 11 show Prism Holdings company's Self Service Terminal system and INFEA corporation's NSA2010 cellular phone, respectively. They are uC/OS products.

Fig. 10 Self Service Terminal.

Fig. 11 INFEA: NSA2010.

7. CONCLUSION

Generally, an embedded System has limited resources. Therefore, every embedded OS must be designed to consider this. This paper analyzes and compares four embedded OSs. Since Widows CE provides very good IDE tools, various OS services, and GUI (Graphic User Interface), it is suitable for designing a high-performance multimedia product. Furthermore, Embedded Visual C or Basic can be used to develop Widows CE application programs. However, Windows CE requires a high-performance processor that are 32-bit or more and must include MMU. Windows CE is not suitable for a time critical application because it cannot support a hard real-time [7].

Embedded Linux supports more processors than Windows CE. For example, embedded Linux supports none-MMU processors, and even supports 16-bit processors. Since Linux source code is open, it is quite flexible. Designers can modify its source code and they can fit its kernel to their own systems. Also, since a lot of people in the world have examined embedded Linux for a long time, it is more stable and secure than Windows CE. However, the development environment of embedded Linux is not as good as Windows CE. There are various versions of embedded Linux, which makes a novice feel hard to apply embedded Linux. Embedded Linux is also soft RTOS, which means that it cannot be used for a time critical application.

Since pSOS is hard RTOS, pSOS is suitable for a time critical application. Compared with Windows CE or Embedded Linux, pSOS requires much simpler hardware and it is easy to develop device drivers. The development environment of pSOS is good. However, pSOS is not suitable for a graphical application because pSOS does not support GUI.

Since uC/OS provides only a kernel code, there are few OS services. Therefore, designers must develop a lot of system programs. However, uC/OS requires the minimum hardware. We can build an embedded system on one-chip micro controller by using uC/OS. Also, since uC/OS is open, it uC/OS has similar advantages as embedded Linux.

As in Table 2, the characteristics of embedded OSs are different from each other. Thus, designers must select an appropriate OS for a desired embedded system [10].

Table 2. The characteristics of each OS.

	Windows CE	Embedded Linux	pSOS	uC/OS
IDE support	YES	NO	YES	NO
Support CPUs	32BIT (MMU)	16bit or Higher	16bit or Higher	8bit or Higher
Memory	350KB	125 - 256KB	60 - 100KB	3-5K
GUI support	YES	YES	NO	NO
Network support	VERY GOOD	VERY GOOD	GOOD	NO
User friendly	VERY GOOD	GOOD	GOOD	POOR
Real Time	SOFT RTOS	SOFT RTOS	HARD RTOS	HARD RTOS
System modification	NO	YES	NO	YES

REFERENCES

[1] Catherine Lingxia Wang, Bo Yao, Yang Yang and Zhengyong Zhu, “A Survey of Embedded Operating System,”

[2] L. M. Thompson, “Using pSOS+ for embedded real-time computing,” *Thirty-Fifth IEEE Computer Society International Conference,* pp. 282-288, 1990.

[3] http://www.windriver.com, *Datasheet of pSOSystem*, Windriver Co., LTD, 2003

[4] Jean J. Labrosse, *MicroC/OS-II The Real-Time Kernel*, Miller Freeman, Inc. USA, 2002.

[5] L. F. Friedrich, J. Stankovic, M. Humphrey, M. Marley and J. Haskins Jr, “A survey of configurable, component-based operating systems for embedded applications,” *Micro, IEEE* , Vol. 21, No. 3, pp. 54-68, 2001.

[6] C. M. Netter, L. F. Baceller, “Assessing the real-time properties of Windows CE 3.0,” *ISORC - 2001. Proceedings. Fourth IEEE International Symposium 2001*, pp. 179-184, 2001.

[7] T. Nakajima, M. Iwasaki and S. Ochiai, “Issues for making Linux predictable” *Applications and the Internet (SAINT) Workshops, 2002, Proceedings,* pp. 8 – 14, 2002.

[8] A. Lennon, “Embedding Linux,” *IEE Review , Vol. 47 Issue. 3, May 2001,* pp. 33 – 37, 2001.

[9] J. H. Lee and H. N. Kim, “Implementing priorit inheritance semaphore on uC/OS real-time kernel,” *Software Technologies for Future Embedded Systems, 2003. IEEE Workshop,* pp. 83-86, 2003.

[10] P. Koopman, “Embedded system design issues (the rest of the story),” *1996 IEEE International Conference, Computer Design: VLSI in Computers and Processors*, pp. 310-317, 1996.

[11] http://www.microsoft.com, *Windows CE product information page*, Microsoft Co., LTD, 2003

[12] http://www.gnu.org, *Embedded Linux page*, Free Software Foundation, Inc, 2003