IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

Comparison of Contemporary Real Time Operating Systems

Mr. Sagar Jape[1], Mr. Mihir Kulkarni[2], Prof.Dipti Pawade[3]

Student, Bachelors of Engineering, Department of Information Technology, K J Somaiya College of Engineering, Mumbai[1,2]

Assistant Professor, Department of Information Technology, K J Somaiya College of Engineering, Mumbai[3]

Abstract: With the advancement in embedded area, importance of real time operating system (RTOS) has been increased to greater extent. Now days for every embedded application low latency, efficient memory utilization and effective scheduling techniques are the basic requirements. Thus in this paper we have attempted to compare some of the real time operating systems. The systems (viz. VxWorks, QNX, Ecos, RTLinux, Windows CE and FreeRTOS) have been selected according to the highest user base criterion. We enlist the peculiar features of the systems with respect to the parameters like scheduling policies, licensing, memory management techniques, etc. and further, compare the selected systems over these parameters. Our effort to formulate the often confused, complex and contradictory pieces of information on contemporary RTOSs into simple, analytical organized structure will provide decisive insights to the reader on the selection process of an RTOS as per his requirements.

Keywords:RTOS, VxWorks, QNX, eCOS, RTLinux,Windows CE, FreeRTOS

I. INTRODUCTION

An operating system (OS) is a set of software that handles computer hardware. Basically it acts as an interface between user program and computer hardware. It provides various services like communications, error detection, program execution, I/O operations, user interface, file manipulation, accounting, protection, security, resource allocation etc. These services are common for all but there are certain applications with specific requirement likeprocessing time. For such applications special OS is designed known as Real Time Operating System (RTOS). The motive behind RTOS development is to process data as it comes in without or minimum buffering delay. Table 1 elaborates different features of a RTOS [1]. Real time operating system is broadly classified in three categories: hard, soft and firm. Fig1 shows the classification of various Real Time Operating Systems viz. Soft, Hard and Firm RTOS [2].

TABLE IFEATURES OF REAL TIME OPERATING SYSTEMS

Synchronization	• Necessary to share mutually exclusive resources. • Use to handle inter-task communication
Interrupt Handling	• Interrupt is handled using Interrupt Service Routine (ISR). • Interrupt latency is zero.
Real-Time Priority Levels	• Support real-time priority levels so that when once the programmer assigns a priority value to a task, the operating system does not change it by itself.
Fast Task Preemption	• Whenever a high priority critical task arrives, an executing low priority task should be preempted immediately keeping delay as minimum as possible.
Memory Management	• provide stack and heap • used during context switching dynamic allocation • Two Types of memory management techniques are there 1. Static Memory management: Memory is allocated at compile or design time. 2. Dynamic Memory Management: Memory is allocated at runtime.

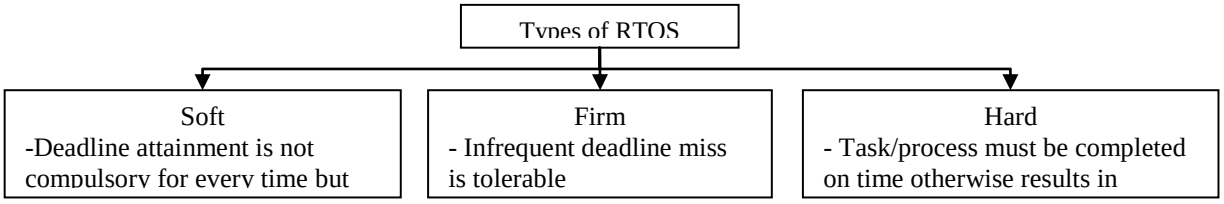

FIG1.CLASSIFICATION OF REAL TIME OPERATING SYSTEMS

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

II. DIFFERENT RTOS

In this section we go further to discuss different real time operating systems and their features.

A. VxWorks

VxWorks is one of the most widely used commercial operating system [4]. It is develop by Wind River of Alameda, California, US and released in 1987. It has following features:

- VxWorks has a multitasking kernel.
- It uses pre-emptive and round-robin scheduling algorithms.
- Fast interrupt response is the biggest strength of VxWorks.
- It supports memory protection mechanism which isolates real time process from other user mode application and kernel.
- Here applications run in kernel mode. Thus all physical memory is accessible to developers in real-addressing mode. This results in highest performance, determinism and flexibility.
- To provide memory protection WindRiver provides bundled, basic MMU support and an unbundled OS extension called VxVMI.
- Designed for: embedded systems requiring real-time, deterministic performance
- Supports: Intel, MIPS, PowerPC, SH-4, and ARM architectures
- It is used in the Mars Reconnaissance Orbiter, the Mars Science Laboratory, NASA Mars rovers, Motorola's DCT2500 interactive digital set-top box, ReplayTV home digital video recorder, Mobile TechnikaMobbyTalk and MobbyTalk253 phones

B. QNX

QNX operating system was released in 1980 by Canadian company Quantum Software Systems. Later that company was renamed as QNX Software Systems and ultimately acquired by BlackBerry in 2010. It is basically designed for the embedded systems with real-time requirement [5]. It has following features:

- QNX is quite small microkernel operating system.
- Its kernel contains only CPU scheduling, interprocess communication, interrupt redirection and timers. Everything else runs as a user process space.
- QNX interprocess communication is carried out as follows: [6]
- MsgSend operation: consists message transfer from one process to another and waiting for a reply
- The kernel copies the message from the address space of the sending process to that of the receiving process.
- If the receiving process is waiting for the message, control of the CPU is transferred at the same time, without a passing through the CPU scheduler.
- QNX Neutrino, released in 2011 was the updated and revised version of the original QNX system; now designed to support Symmetric Multiprocessing and Processor affinity. The revisions also provided Support towards the then latest POSIX APIs and also to the API versions which could have been then anticipated.
- An exception is found in QNX Neutrino scheduling. Here threads are scheduled not the process. For scheduling purpose thread priority is considered.
- Supports: PowerPC, x86 family, MIPS, SH-4, and the closely inter-related family of ARM, StrongARM and XScale CPUs.

C. eCos

eCos stands for Embedded Configurable Operating System. It is introduced in 1997 by Cygnus Solutions. Name of this RTOS is self-explanatory saying it is appropriate for embedded systems which desire to build up as per the application specifications and need only one process with multiple threads. Its peculiarities are given below: [7]

- eCos is a non-proprietary customizable real-time operating system.
- Support following schedulers
- The bitmap scheduler: most suitable for system with small number of threads. This scheduler has lowest execution overhead. It supports fixed number of threads.
- The multiple priority queue based scheduler: suitable for system with dynamic number of threads. It is also useful when time slicing is required.
- Supports: ARM, CalmRISC, FR-V, Hitachi H8, IA-32, Motorola 68000, Matsushita AM3x, MIPS, NEC V8xx, Nios II, PowerPC, SPARC, and SuperH.[11].

D. RTLinux

- RTLinux is also known as real time extension of Linux. RTLinux is developed by Victor Yodaiken, et al. at the New Mexico Institute of Mining and Technology and then as a commercial product at FSMLabs. Wind River Systems acquired FSMLabs embedded technology and launched its version as Wind River Real-Time Core for Wind River Linux. It is designed foe time sensitive embedded system [8]. Its characteristics are as follows:
- It is a hard RTOS
- It has preemptive microkernel that runs the entire Linux operating system as a fully pre-emptive process.
- It has two parts
- Real time : runs on the RT kernel
- Non-real time: runs on Linux.
- The communication between the real-time and non-real time component is via FIFO buffers, called RT FIFOs.
- Has many kinds of schedulers:
- FIFO: Used to pass information between real-time process and ordinary Linux process.
- Earliest Deadline First scheduler.
- Rate-monotonic scheduler
- Supports: Alpha, ARC, ARM, AVR32, Blackfin, C6x, ETRAX CRIS, M32R, m68k, META, Microblaze, MIPS, MN103, Nios II, OpenRISC, SPARC, x86[11].

E. **Windows CE**

Microsoft announced Windows CE at the COMDEX expo in 1996. It was demonstrated on stage by Bill Gates and John McGill [9]. It supports x86, MIPS and 32-bit ARM platforms. It is especially designed for time sensitive embedded systems. It has following features:

- Windows CE is designed for devices that have minimal memory; one megabyte of memory is enough for a Windows CE kernel to run.
- Devices are often configured without disk storage, and sometimes may be configured as a "closed" system that prevents end-user extension (for instance, it can be burned into ROM).
- The deterministic interrupt latency of Windows CE confirms it as a real-time operating system.
- A feature of Windows CE which makes it distinct from other Microsoft operating systems is that large parts of it is available in source code form. Earlier, several vendors were offered source code, so that they could adjust it to their hardware. Later products such as Platform Builder (an integrated environment for Windows CE OS image creation and integration, or customized operating system designs based on CE) provided the source code of several components to the general public. However, a number of core components that are independent of specific hardware environments (other than the CPU family) are still distributed in binary only form.

F. **FreeRTOS**

- FreeRTOS is a free to embed open source real time operating system which supports about 35 microcontroller architectures. The FreeRTOS project was founded by Richard Barry from Real Time Engineers Ltd. which are the owners and maintainers of the project. FreeRTOS started circulation in 2003; its latest version is 8.2.2 released on 12th August 2015. Its features include:
- It uses a perculiarly efficient software timer implementation. The timer does not consume any CPU cycles and resources unless it invoked for servicing [Ref:www.freertos.org/RTOS]
- FreeRTOS Queues used for Interprocess Communication are significantly different from other RTOS. These queues used data passed by values instead of passing by reference (Queue stores the data itself instead of pointers to the data). This ensures simplicity and flexibility. [10]
- Free RTOS supports tick-less mode for low power-applications.
- It supports multithreading using priority based round robin scheduling. The host program calls the 'thread tick' method at regular intervals. The thread tick method does the job of task switching.[12]
- It is has a minimal ROM footprint (6K to 12K). It is written mostly in C with the kernel comprising of just three to four C files.
- Provides primitive memory management techniques like allocate and free algorithms with memory coalescence. It also provides C libraries which contain functions for trivial mutual exclusion options.[10][11]

TABLE II COMPARISON OF VARIOUS REAL TIME OPERATING SYSTEMS

RTOS	**License**	**Scheduling Algorithm**	**Platforms**	**Memory Allocation**
VxWorks	Proprietary	Preemptive and Round Robin Scheduling	ARM, IA-32, Intel 64, MIPS, PowerPC, SH-4, StrongARM, xScale	Best Fit Algorithm
QNX	Proprietary	Priority-Preemptive Scheduling	IA-32, MIPS, PowerPC, SH-4, ARM, StrongARM, XScale	Strict Memory Protection by Memory Management Unit
eCos	Modified GNU GPL	Bitmap Scheduler and Multiple-Priority, Queue-Based Scheduler	ARM-XScale-Cortex-M, 680x0-ColdFire, fr30, FR-V, IA-32, MIPS, MN10300, OpenRISC, PowerPC, SPARC, SuperH	Memory Pool Based Dynamic Memory Allocation
RTLinux	GNU GPL	FIFO, Earliest Deadline First Scheduler	Alpha, ARC, ARM, AVR32, Blackfin, C6x, ETRAX CRIS, M32R, m68k, META, Microblaze, MIPS, MN103, Nios II, OpenRISC, SPARC, x86	Uses Regular Linux Memory Management Provisions. No Real Time Allocation
WinCE	Proprietary	Priority-Based Time-Slice Algorithm	ARM, MIPS, SH4 and x86 Architectures	Large Memory Mapped File Support
FreeRTOS	Modified GPL License	Priority Based Round Robin Scheduling	ARM (ARM7, ARM9, Cortex-M3, Cortex-M4, Cortex-A), Atmel AVR, AVR32, HCS12, MicroBlaze, Cortus (APS1, APS3, APS3R, APS5, FPF3, FPS6, FPS8), MSP430, PIC, Renesas H8/S, SuperH, RX, x86, 8052, Coldfire, V850, 78K0R, Fujitsu MB91460 series, Fujitsu MB96340 series, Nios II, Cortex-R4, TMS570, RM4x	Primitive Allocate and Free Algorithms with Memory Coalescence.

DOI 10.17148/IJARCCE.2015.41192

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

IV. CONCLUSION

In this paper we have discussed various real time operating system like VxWorks, QNX, eCOS, RTLinux,Windows CE, FreeRTOS. We tried to highlight their salient features and usability. From our survey we conclude that:

- Microkernel architecture of QNX makes it more robust.
- RTLinux extends the capabilities of Linux kernel to have real time computation power and hence justify its claims of being a hard real time operating system.
- VxWorks is known to be the most successful commercial RTOS and has fastest interrupt response.
- Feasibility to configure is the strength of eCOS.
- Windows CE is also considered to be a robust RTOS with number of platforms having variety of utilities.

REFERENCES

[1] DiptiDiwase, Shraddha Shah, TusharDiwase, PriyaRathod, "Survey Report on Memory Allocation Strategies for Real Time Operating System in Context with Embedded Devices", International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, May-Jun 2012, pp.1151-1156
[2] Jane W. S. Liu, "Real-time System", published by Person Education, first edition
[3] Nandana V. JithendranA,Shreelekshmi R., "Survey on RTOS: Evolution, Types and Current Research",International Journal of Computer Applications (0975 – 8887) Volume 121 – No.21, July 2015,pp28-31
[4] http://windriver.com/products/vxworks/
[5] http://www.qnx.com/
[6] Rodrigo CarvalloCroske, "QNX Microkernel based OS", publisher Pediapress.
[7] http://ecos.sourceware.org/about.html
[8] www.rtlinux.org/
[9] https://msdn.microsoft.com/en-us/library/aa450610.aspx
[10] http://www.freertos.org/Embedded-RTOS-Queues.html
[11] Sanjay Deshmukh, Dr. N. N. Mhala, "Comparison of Open Source RTOSs Using Various Performance Parameters" International Journal of Electronics Communication and Computer Engineering, Volume 4, Issue (2) REACT-2013.
[12] www.freertos.org/RTOS

DOI 10.17148/IJARCCE.2015.41192