Real Time Operating Systems: A Review

Kanak Priya[1] Atul Kumar Srivastava[2]
[1]Student [2]Asst. Professor
[1, 2]AITEM, Noida

Abstract--- A real-time operating system is software system which is designed to ensure the time critical events are processed as efficiently as possible and must possessed characteristics like guaranteed calculation, low latency and have deterministic behavior which is not applicable for general purpose operating system or standard operating system and also general purpose operating system failed to give real time performance. This paper aims to do a survey study and explains such real time operating systems.
Keywords: GPOS, Linux, OS, RTOS

I. INTRODUCTION

An operating system can be defined as the system software to provide the user interface to control the activities of computer hardware. When the machine is powered on, it is the first program to be loaded onto the computer memory. Some examples of the operating system are MS DOS, Linux, Windows 98, and Windows XP etc. The aim of OS is to provide services like input/output, multitasking, interrupt and event handling, Inter-task communication, memory management etc. The hearts of an Operating System are its kernels which are optimized set of libraries [1].

RTOS or the real time operating system supports real time applications by providing logically correct result within the dead line as required [3]. Basic structure is similar to standard operating systems; however it additionally provides mechanism to allow real time scheduling of tasks. It may not increase the quantity or speed of execution, but it does provides more precise and predictable timing characteristics than general purpose or standard operating system based upon degree of tolerance in meeting deadlines**.** There are two types of RTOS classifications [4]

A. Hard real time

This is a class of RTOS in which degree of tolerance for missed deadline is negligible. A missed deadline can result in catastrophic failure of the system.

B. Soft real time

This is a class of RTOS in which deadline may be missed occasionally, but system doesn't lead to fail.

For example, imagine that some engineers developed an automatic censor based system for landing gears of aero planes. Now if the landing gears don't come out in real time, the situation will be catastrophic causing death or injury to so many passengers on board. What if the landing gears open after the plane hits the ground. Such applications need hard real-time systems. On the other hand, a soft real-time operating system may suffice for lesser important things like video calling software [5][6]. Even if the video doesn't stream in real time it won't do any harm.

II. RTOS

In this section, we will explain how RTOS kernels can be generated with reference to Linux Operating system. There are two basic approaches by which a standard Linux kernel can be modified into RTOS Linux kernel [7] [8].

- *Micro-Kernel:* This approach introduces a new and highly efficient code layer between the hardware and standard kernel. This additional code layer is called as micro-kernel; take care of the entire of the real-time functionality that includes interrupts, scheduling, and high resolution timing. This micro-kernel layer runs the standard kernel as a background task. An alternative code layer, called as a nano-kernel is the one that only handles interrupt dispatching. This nano-kernel approach can support multiple operating systems with one as real time OS and others as standard [9].
- *IEEE 1003.1d Kernel:* This second approach is to implement the real-time extension to POSIX.1 within the configuration of the standard Linux kernel. With these extensions timer, scheduling, and pre-emption logic are directly added into a single, monolithic Kernel. The standard kernel in current developer releases shows latency and jitter of order of one millisecond**.** The real-time versions of Linux contain latency and jitter of the order of a few microseconds on the processors running at the several hundred megahertz [10]. This approach is one very advanced and common approach.

Standard Operating systems like Windows are designed to the maintain better user responsiveness by means of multiple programs and services while the real-time operating systems is one with both the logical correctness of the outputs as well as their timeliness [11][12]. A real-time system that must satisfy bounded response-time constraint; otherwise risk severe consequences, includes failure. An RTOS differs from GPOS, in that the user when using the former have the ability to directly use the microprocessor and peripherals devices. Such type of ability of the RTOS must helps to meet the deadlines [12].

III. DIFFERENT AVAILABLE RTOS

There are different RTOS available and they are [13] [14] [15] [16]:

A. Free RTOS

This is one of the simple and the famous real time operating system and is written in C language. Such famous and simple RTOS basically uses simple and small kernel and make operating system compatible with the embedded system. And also revealed to have low overhead. Attribute priorities the scheduling policies like fixed-priority pre-emptive scheduling and round robin (RR) for threads having similar priorities [13].

B. Enea OSE

Enea OSE is developed by a Swedish firm, Enea AB. This is a compressed RTOS which is widely used in the mobile phones. Such OS are compatible with ARM Platform. Enea OSE multicore was releases in 2009 based on the same kernel architecture as Enea OSE. Task attributes like: priority scheduling policy: fixed-priority pre-emptive scheduling, periodic scheduling, round robin scheduling for task having the same priority [14].

C. QNX

QNX is a kind of RTOS which mostly aims embedded systems. This was first released in 1982 by the research. In motion is Unix-like OS which generally executes most of the operations in form of small tasks i.e. known as servers. Task attributes like: no WCET or non-functional attribute specifiable scheduling policies. They are the round robin scheduling, FIFO scheduling and adaptive scheduling [15].

D. Lynx OS

Lynx OS is an RTOS written in C, C++ and Ada language which had its very first release in 1986 and currently the fifth version of operating system is available for the users. It is a closed source operating system. Undemanding priority based scheduling publishes having accurate real-time performance ,and number are in accordance with the system's performance in real –world condition such as context switching time, pre-emption time, interrupt latency, and many others. Scheduling policy like: fixed priority pre-emptive scheduling [16].

E. VxWorks

VxWorks is a 64-bit RTOS i.e. built for the embedded systems. Its support shared resources services as well as local and distributed messaging queue and benefit from a compressed design have made this RTOS useful in different industries working with embedded systems, like airplane and the vehicle production. Task attributes like: priority scheduling policies: pre-emptive fixed priority scheduling (256 priority levels), Round robin and also use POSIX i.e. FIFO and RR [17].

IV. FEATURES OF RTOS

A real time system involves the logical correctness of the output and accuracy of time. It follows the bounded time response constraint or else it may result in a severe failure. Some desired features in an RTOS may include the ability to schedule a task and meeting deadlines, easy incorporation of the external hardware and error recovery, fast switching among the tasks, small size with overheads. Following are the requirements of an RTOS [18].

A. Multithreading and Pre-emptibility

An RTOS should be multi threaded so that multiple tasks can be run simultaneously in real time and also the scheduler should be able to pre-empt any of the thread in the system and also gives the resource to the thread that desires it most. Similarly multiple interrupts should also be handled by an RTOS [18] [19].

B. Thread priority

An RTOS should be able enough to determine the threads which need fewer resources and the threads which need more resources. To handling deadline each thread is assigned priority level. Deadline information is converted to priority level of threads. Although the approach of resource allocation among competing thread is prone to error, in absence of another solution, the notion of priority level is used in an RTOS [18].

C. Predictable thread synchronization

A predictable inter thread communication and synchronizing mechanism is required for threads to communicate among each other, in a timely manner. Also, the supported must the ability to lock/unlock resources to reach data integrity [17].

D. Memory management

RTOS which are designed for the large and the medium sized applications usually provide virtual memory support, not only to meet up the memory demands of the heavyweight, real-time tasks of an application, but also to hire the memory demanding of non-real-time applications like text editors, e-mail etc. An RTOS always uses small amount of memory by taking only the required functionality for an application while it ignore the rest [18].

E. Priority inheritance

There should be some priority levels in RTOS, so that the applications with stern priority requirements can be implemented. There might be a condition when whenever a higher priority task should wait for a lower priority task to release recourses, while the low priority task is waiting for the medium priority task. This condition is known as priority inversion and an RTOS must be able to present this. In this case the blocking task is capable of finish execution without being pre-empted by the medium priority task. The designer must make sure that the RTOS used must prevent such priority inversion [17] [18].

F. Predefined latencies

An OS supporting the real-time applications desires information about the timing of its system calls. The behaviour metrics to be specified as below [19][20]:

1) Task switching latency

It is the time to save the context of a currently executing task and switch to next task.

2) Interrupt latency

It can be defined as the time difference between the execution of the last instruction of the interrupted task and the first instruction in the interrupt handler condition. This is a metric of the system that response to an external event.

3) Interrupt dispatch latency

This is actually the time to go from last instruction in the interrupt handler to the next task scheduled for run. This indicates exactly the time needed to go from interrupt level to task level.

V. RTOS KERNEL

A Kernel is the core part of an operating system which is actually responsible for connecting the hardware of the system to the application layer. RTOS is mainly responsible for task like creation, execution and deletion according to the policy set by programmer. Also general purpose kernels, the tasks of memory management, interrupt handling and managing input/output communication are perform by the

RTOS kernel. However the main difference between an RTOS kernel and a general purpose kernel is that the non-real time kernels never assure that they will perform real-time application in a given timing. Following are the main contents of a kernel .

A. Scheduler

It manages role in the kernel i.e. allocating processor time to all the tasks based on the scheduling policies which are basically designed for it [19].

B. Interrupt handler

Its aim is to manage the request coming from a different hardware tool as well as system calls. An interrupt handler can be either initiated by interrupts coming from the hardware or by executing the software integrated interrupts codes [19].

C. Memory manager

This section provides location in system memory and is known as memory allocation. An RTOS provides two types of memory management services and they are stack and heap where they are used during the context switching for task control blocks and with memory other than memory used for program code and used for dynamic allocation of data space for tasks respectively. RTOS also support static and dynamic memory management. Static memory allocation is deterministic process that means that the required memory for particular process which is already known and once memory is allocated then after that no changes can be done at the time of running process. But dynamic memory manager requires memory manager that tracks which part of the memory is used and which are still unused. This helps to allocate memory to the processes when they need it and deal locate when they done [20].

VI. TASKS

A Task represents a single as well as the multiple type of process that an operating system is required to execute. A process is a single occurrence of a task whose function is to receive necessary resources from the operating system and a job mainly consist of set of tasks. In real time, the timing constraints like release time, period execution or computation time, response time, deadline are assigned to a task. Some of the basic concepts used in real-time system analysis are explained below [20][21] :

- *Release Time:* It is the time period during which task get for scheduling.
- *Period:* It is the time span after which the next task instance is released.
- *Execution Time:* It is the period of time spent by the system while executing a task.
- *Response Time:* It is the time period between a task free and its execution to completion.
- *Deadline:* It is a very important parameter. There are two common terms regarding to deadline in case of real-time analysis. They are absolute deadline and relative deadline.
- *Periodicity:* Periodicity is the execution of task instances within the fixed time interval.
- *Task state:* In case of task state, the task in the waiting line is in suspended state. Three general states that a task could have are: ready, running and suspended.
- *Ready state:* In this an OS received all the resources and the data which is needed for the execution, and waits for the scheduler to allow it run in its assigned CPU time.
- *Running state:* When the time reaches for the task to get it executed; it is placed in running state.
- *Suspended state:* Suspended state is a state, when it is waiting for the shared resources to be the available, or an event to happen.

VII. APPROACHES TOWARDS AN RTOS

In real life a number of solutions have been suggested for making Linux compatible with the real-time environment. In this section an overview of the most common solution for improving the real-time performance of the Linux operating system. More details can be find through the references [23] [24].

- Interrupt Abstraction
- RTLinux
- RTAI
- Xenomai
- Reducing the Latencies
- Pre-emptive Kernel Patch
- Low latency Patch
- Pre-empt rt Patch

VIII. TECHNICAL ADVANTAGES OF RTOS

Following are the advantages of an RTOS

A. Performance

The performance bench mark setup by the real time Linux is far better than most of the currently existing high performance proprietary RTOS. Also the technical concurrency of Linux with major stream software engineering development is at least similar. If not being superior to other operating system, it is typically at top edge of software engineering concept for those prepared to utilize the odd number releases [15] [16].

B. System Services

When comes to deal with the system services then real time Linux basically brings all the services and complicated of a typical Linux operating system. Accomplished by symbiotic connection in which Linux provides the general operating system services, while the real time extension afford the time-critical services [17].

C. Maintainability

While comparing with the other proprietary alternatives RTOS, with the growth in significant rate, the standard Linux is still being the most compressed and handy operating system [15].

D. Reliability

Linux is one of the secure OS today. For over a year without getting intermittent; there are number of servers operating endlessly. Freshly a real time extension is dealing with the different story and this is always lacking the main kernel development. The conditions are improved these days with

the increase in the number of real users and commercially support testing and justification [15].

E. Customization

It is possible to modify the kernel of such OS to create a compressed stripped down kernel for any particular application. For the embedded processor board computer firm this is the market that aiming for WINCE. In the interests of full debate, it should be concluded that the customizing process for the real-time kernel is not the GUI based relations i.e. being afforded by WINCE but in the aegis of a competent authority that may accomplished reliably, affordably within a few calendar weeks [16].

F. Platform Availability

Linux is available on Intel's platform. After the release of a stable 2.2 versions, the numbers of hardware vendors have expressed their interest in version for the Alpha, strong Arm and 68000. Subscription mechanism was developed by one of vendor [16].

REFERENCES

[1] Zhang G., Chen L., and Yao A. Study and comparison of the rthal-based and adeos- based rtai real-time solutions for Linux. In First International Multi-Symposiums on Computer and Computational Sciences, IMSCCS, pages 771-775, June 2006.
[2] Mantegazza P. Dissecting diapm rthal-rtai. http://www.aero.polimi.it/rtai/ documentation/articles/paolo-dissecting.html. [Online; accessed October- 2013].
[3] Anderson M. P. and Lindskov J. Real-time Linux in an embedded environment. Master of Science Thesis, Lund University, January 2003.
[4] Regnier P., Lima G., and Barreto L. Evaluation of interrupt handling timeliness in real-time Linux operating systems. In ACM SIGOPS Operating Systems, pages 52-63, October 2008.
[5] R. Love. Lowering latency in Linux: introducing a preemptible kernel. Linux Journal May 2002.
[6] Ingo Molnar, Linux low latency patch for multimedia applications. http://people. redhat.com/mingo/low latency-patches/. [Online; accessed October-2013].
[7] A. C. Heursch, D. Grambow, A. Horstkotte, and H. Rzehak.Steps towards a fully preemptable Linux kernel. 2003.
[8] C. Scordino and G. Lipari. Linux and real-time: Current approaches and future opportunities. In ANIPLA International Congress, Rome, 2006.
[9] Real-Time Linux – What was/is expected?(Assignment :6 Patel, Shital (900-01-9649))
[10] A.Arnaud and I.Puaut.Towards a predictable and high performance use of instruction Caches in hard real-time sys-tems.In Proc.of the work-in-progress session of the 15th Euromicro Conference on Real-Time Systems, Porto,Portugal,July.003,61~64
[11] .A.Arnaud and I.Puaut.Dynamic instruction Cache lock-ing in hard real-time systems. In Proc.of the 14th International Conference on Real-Time and Network System (RNTS),Poitiers,France,May 2006,32~37
[12]. MS Johnstone,"Fragmentation Problem: Solved?"In Proc. of the Int. Symposium on Memory Management (ISMM'98), Vancouver, Canada. ACM Press
[13] .S. Baskiyar, Ph.D. and N. Meghanathan, "A Survey of Contemporary Real-time Operating Systems" Informatics 29 (2005),Pg No.233–240
[14] Jane W. S. Liu, "Real-time System", (published by Person Education, Page no. 20-40).
[15]. Robart L. Budzinski, Edward S. Davidson, "A Comparison of Dynamic and Static Virtual Memory Allocation Algorithms" IEEE Transactions on software Engineering, Vol. SE-7, NO. 1, January 1981.
[16] .Introduction to Linux for Real-Time Control,NIST: Intelligent Systems Division ,Version 2.0.0, 2002-12-11T153117.A Linux based Real Time Operating System
[17] .A Linux based Real Time Operating System by Michael Barabanov, New Mexico Institute of Mining and Technology,june 1,1997
[18]. Intel Corporation. Intel 486 (TM) Processor Families. Programmers Reference Manual 1995. Order Number 240486-003.
[19]. Alan Burns. Scheduling hard real time systems: A reviews software engineering Journal, 6(3): 116-128, 1991.
[20] .Linux Kernel Development, Third Edition. By Robert love; chapter-4.
[21] Sang H. Son editor .Advances In Real-Time Systems, chapter 10, page 225-248, prentice Hall, 1984.

