Comparison between scheduling algorithms in RTLinux and VxWorks

Linköpings Universitet
Linköping

2006-11-19

Daniel Forsberg (danfo601@student.liu.se)
Magnus Nilsson (magni141@student.liu.se)

Abstract

The scheduler decides wich one of the available tasks that will be next to be processed by the CPU. To make this decision the scheduler checks the priority levels of the scheduled tasks and apply a predefined schedule algorithm. This report will compare the scheduling algorithms in the real-time operating systems RTLinux and VxWorks and comes to the conclusion that the two compared systems are roughly equivalent in most uses in regards to this.

Table of Contents

1 Introduction

While many non-real-time operating systems try to implement realtime behaviour (sometimes only to get the "real-time"-label on their product) none or few of them get the response times required for a hard real-time system. A hard real-time system is a system that can assure task completion within its given time constrains. A system that can't guarantee task completion before deadline but still gives real-time tasks priority over ordinary tasks is called a soft real-time system. The entity "task" is used throughout this text as both RTLinux and WxWorks sees processes and threads as tasks. RTLinux and VxWorks are two of the most videly used and successful hard real-time operating systems on the market today and therefore makes good examples of how efficient scheduling can be implemented. In section X we discuss the RTLinux os, in section Y we dvelve into the VxWorks RTOS and our conclusions are found in section W.

2 Scheduling techniques

2.1 Round robin scheduling

Round-robin scheduling gives every process with the same priority a pre-set share of time before making a context switch to the next task. When all tasks have got their time share, the first task gets back into the CPU for it's next processing.

2.2 First-in-first-out (FIFO) scheduling

The scheduler runs the task with the highest priority first. If there are two or more tasks that share the same priority level, they get scheduled in order of their arrival completing the first arrived task first before continuing with the next one. Each task is occupng the CPU until it finishes or another task with higher priorty arrives.

2.3 Earliest-deadline-first (EDF) scheduling

This scheduling policy ignores the priority level for each task. Instead it focus on when each task has to be finished, choosing the task with the closest deadline for execution. The more accurate the provided deadlines are, the better CPU utilization can be expected.

2.4 Rate-monotonic scheduling

The rate-monotonic scheduling algorithm sets the priority level for each task in order of their period length, tasks with short periods (they executes often) will get a high priority while tasks with long periods gets low priority. High priority tasks then take preceedence before lower priory tasks. This scheduling algorithm is best used if there are well defined periodic tasks, prefereable with the same CPU burst length (the time spent in the CPU for each instance of the task).

3 RTLinux

RTLinux by FSMLabs is a commercial hard realtime operating system that is used in many embedded systems worldwide. Instead of using the preemptive kernel of Linux for real-time applications, RTLinux uses a specially designed realtime kernel

called RTCore. This means that RTLinux actually is running two kernels, both the real-time kernel for time-critical applications as well as a normal Linux kernel for regular applications without time constraints. All interrupt handling and thread scheduling is controlled by the RTCore wich directs each interrupt to the appropriate interrupt handler, either in the real-time kernel or the standard Linux kernel. RTCore also prevents the Linux kernel from disabling interrupts, making sure that Linux doesn't interfere with the RTCore scheduling. The Linux kernel is only allowed to run when there is no current real-time demands - it is treated as a lowest level priority task by RTCore. Interrupts to the Linux system are in fact emulatetd interrupts as they allready passed the RTCore interrupt handler. Real-time applications are able to communicate with Linux programs through FIFO pipes and/or shared memory. The POSIX (Portable Operation System Interface eXchange) API threading functions are implemented in RTCore, lowering the learning curve for programmers allready familiar with this API.

From the RT kernel's point of view, the regular Linux kernel is just another task and this allows it to give higher priority to real time applications. From the user's point of view, this means that a system running RTLinux has all the available functionality of an ordinary Linux while still providing full real time capability.

3.1 Scheduling in RTLinux

RTLinux can use different scheduling algorithms depending on the needs of it's users. RTLinux uses a simple FIFO scheduling but can also use EDF and rate-monotonic scheduling if so desired. For compatibility reasons RTLinux makes available the POSIX function calls for scheduling, but ignores those calls.

4 VxWorks

VxWorks by Wind River Systems has been on the market for many years, evolving as the market's needs have changed over time. In contrast to RTLinux, WxWorks runs only one kernel - the Wind microkernel. This microkernel handles only the most basic kernel functions, which means that additional features like file handling, networking, etc, are loaded from provided libraries as needed. This gives high flexibility when designing a system, as one can set the system up to fit the needs of functionality while keeping strict constraints on available memory and other resources. The WxWorks kernel uses POSIX API and the Pthread API for real-time threads and processes.

4.1 Scheduling in VxWorks

All work carried out in VxWorks is in the form of tasks. Each task can be in one of four different states; ready, delay, pending and suspend state. Ready tasks are those tasks available for running. Delayed tasks are sleeping for a set amount of time. Pending tasks are waiting for a resource to be available. The suspended state is what newly created tasks are set to until they get activated, something which however usually is done when the task is created, so this state is primarily used for debugging purposes.

VxWorks uses a priority based preemptive round-robin scheduling algorithm. Each task have a priority level between 0 and 255 where 0 is the highest priority and 255 the lowest priority. If a task with higher priority than the task currently running in the CPU is called, the scheduler suspends the first task and sets the CPU to running the second task. If the second task has the very same priority level as the first task, round-robin scheduling is provided. As mentioned earlier, a task can also turn into a pending state, for example if a resource for that task isn't available. The scheduler then swaps back to a task with lower priority until the resource becomes available again. VxWorks also supports the POSIX API for real time threads, which makes available both the above described round robin as well as FIFO scheduling which is described above in the RTLinux-section. VxWorks scheduling is thus flexible and adopts easily to the needs of the customer.

5 Conclusions

Both RTLinux and VxWorx handle several different methods of task scheduling, making them both candidates for several different software projects. Only VxWorks support the functionality in the POSIX API for threads, VxWorks uses a priority based scheduling algorithm and can use this to implement both a round-robin scheduling as well as a first-come-first-served scheme, while RTLinux may use diferent scheduling algorithm's such as earliest-deadline-first or rate-monotonic scheduling, but also the first-come-first-served that VxWorks can use. Thus, both systems are fairly flexible and if a FIFO or priority based scheduling is desired, the choice of which system to use should not be based upon the desired scheduling method, as both can handle this in the desired way.

To conlude, RTLinux has EDF and Rate-monotonic scheduling available, so if either of these are wanted, RTLinux should be the chosen system while VxWorks would be the nessecary choice if a round-robin scheduling is needed. In all other cases, they are equivalent as far as scheduling goes, and decisions must be made on other merits of the two systems.

6 References

Silberschatz, Galvin and Gagne: Operating System Concepts, 7th ed., John Wiley & Sons, 2005.

http://windriver.com

http://rtlinux.org