A Survey of Operating System Support for Real-time Applications

University of Columbia
Computer Science Department
Dirk Bridwell dnbridwell@earthlink.net
Jan Miller janalanmiller@uswest.net

Abstract

Operating systems' (OS) support for applications, which require real-time behavior, are growing in number and diversity. From the desktop to very specialized software/hardware combinations, developers are required to create systems that can satisfy both hard real-time and soft real-time requirements. Since design cycles can be so short, and projects are becoming more diverse in their demands, developers want familiar and robust Application Programmers Interfaces (API), tools, and programming environments to create their real-time applications. The Developers' ability to integrate with their familiar tools is important, but popular operating systems do not have equal levels of support for real-time applications. The design of an OS can have a significant impact on its ability to be used in a real-time system. Some operating systems support only soft real-time, others support hard real-time, and some support hard real-time with a few caveats. This survey should help developers understand some of the tradeoffs when picking an OS for system design. We will compare the popular operating system Windows NT/2000, two flavors of Linux (plain Linux and RTLinux), and one less known operating system named sPSOS+. Real-time aspects of the operating systems' APIs and the underlying implementation that supports those APIs will be discussed.

Introduction

There are several requirements for an operating system must fulfill to be considered viable candidates for real-time systems. Since most real-time systems need to react to external events and control asynchronous devices, the OS must be multithreaded. Threads (or their conceptual equivalent) are an essential part of a real-time capable OS. Without threads of some kind, the application would not only have to make all the decisions about what should run when, but also implement this functionality. The priority of a particular thread must always be able to take precedence over any other system activity. High priority threads in the system must be able to run before or in the middle of low priority threads, so a candidate system must also be preemptible in order to be deterministic. If the system is going to scale or do anything involving much complexity, the OS has to support multiple thread priorities. Thread synchronization can create many complexities in and of itself, which can affect real time performance and determinism, so predicable synchronization mechanisms need to be implemented by the OS. An OS that supports priority inheritance will prevent priority inversion by supporting priority inheritance mechanisms. Without priority inheritance mechanisms a low priority thread could easily prevent a high priority thread from getting its work done in time to meet a deadline. The operating systems considered in this survey have varying degrees of support for the essential elements of real-time systems.

References

RTLinux-3.0 Documentation: Finite State Machine Labs, Inc., 914 Paisano Drive, Socorro, NM 87801, ftp://ftp.rtlinux.com/pub/rtlinux/v3/ rtldoc-3.1.tar.gz

The RT-Linux Approach to Hard Real-time: Victor Yodaiken, Department of Computer Science, New Mexico Institute of Technology, Socorro, NM

POSIX in Real-Time: By Kevin M. Obenland, Embedded Systems Programming, (03/15/01, 04:00:11 PM EDT), http://www.embedded.com/story/OEG20010312S0073

Linux for Real-Time Systems: Strategies and Solutions, Kevin D. Morgan, MontaVista Software, http://www.techonline.com/scripts/tol.exe?TEMPLATE,top.ops&AREA,41&CONTENT,8795&NET,0&USER,jan_miller@yahoo.com

Real-time Support in General Purpose Operating Systems: Kartik Gopalan, Computer Science Department, State University of New York at Stony Brook, Stony Brook, NY 11794-440.