Implementation of a Signal Control System in a Real Time Environment

Vanessa Frías-Martínez
Computer Science Department
Columbia University
110 Amsterdam Avenue, New York, NY 10027
e-mail: vf2001@cs.columbia.edu

***Abstract* – Real time applications have time requirements to be accomplished. The missing of those deadlines make the system incur in fatal errors. Many operative systems(OS) have been upgraded to achieve preemption in order to offer applications the capability of meeting their deadlines. Linux, with its open source philosophy and robustness offers the opportunity of kernel code manipulation, as well as straight forward testing with the modules idea. This paper presents a real-time application implementation using a real time Linux kernel. The main goal is that of testing how well the real time OS does in confront to a non-preemptive OS. The real time application will control the lighting of eight leds designing models in a 3D space.**

1. INTRODUCTION

A real time system is one capable of guaranteeing timing requirements of the processes under its control. It must be fast and predictable. Fast meaning low latency, that is, responds to external asynchronous events in a short time. The lower the latency, the better the system will respond to events which require immediate attention. Predictable so as to be able to determine the task's completion time with certainty. A typical real time task will have timing constraints, resource requirements, communication requirements and concurrency constraints, all of which has to be treated.
Linux is a POSIX 1003 compliant OS. Processes can be locked into memory to prevent being paged to hard disk. But a Linux kernel does not provide the required event priorization and preemption functions needed by a real time process.
[2][6] and [9] deal with the implementation of real time Linux kernels. The preemption improvements approach, makes modifications in the original native Linux code so as to reduce the time spent by the kernel in non-preemptive sections of code. But, this approach only affords soft real time since when Linux interruptions are disabled by processes, no effective response is guaranteed. The interrupt abstractions approach defines a two layer system where the standard Linux is run as a low priority process along with all the RT high priority preemptive processes, that are run in kernel space. The RT kernel will handle all the interruptions directly.

II. RELATED WORK

Linux distributions differ not in the main kernel, which is common and unique for all of them, but in the applications they include, their graphical environments and other extras.
Open source implementations are being used by companies to offer complemented-payable-kernels coupled with a varied range of tools. The open source philosophy makes Linux world more active and rich in developments.
Some real-time open source Linux kernels, like RTLinux[9] or RTAI[2], work on the philosophy of modules. Its main feature is that RT processes are considered as loadable modules. Those OS achieve response times of 15 µsecs. This is obtained through running the RT processes in the kernel space. In order to avoid possible memory intrusions, both offer the possibility of user space execution. Both real time versions emulate standard Linux interrupts enable/disable so as to avoid the priority inversion problem between non-preemptive and preemptive RT processes.
[5] offers a more detailed and clear two layer system description known as HAL (Hardware Abstraction Layer). It supports five core loadable modules which provide the desired on-demand, real time capability. Those implement the scheduler, memory sharing, clocks control and FIFOs implementation. This will be our base system.

III. SIGNAL CONTROL SYSTEM IMPLEMENTATION

[1] and [3] offer descriptions of real time modules ranging from the simple control of a port dealing with real time signals to the whole implementation of the control for a PUMA robotic arm, including identification of RT threads and RT Linux processes.
Our real time signal control system will develop a loadable module for the RTAI kernel[2]. The application will take control of a real time input complex signal coming from the serial port and will implement a feedback control on it. The actual system will control the lighting of eight leds designing models in a 3D space.
[7] does a comparative study of the output of an audio process (soft real time) in a preemptive and non-preemptive OS, pinpointing the failures in the last one. We will follow a similar study on the output of our system.

IV.REFERENCES

[1] Andris, P. Robot Control using Real Time Linux. In International Workshop on Robotics, Slovenia, 2000.
[2] Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, RTAI, www.rtai.org
[3] Kupper, J.The serial port driver of real time Linux. Institut fur Physicalysche Chemic, Frankfurt, 2000.
[4] Lineo, www.lineo.com
[5] Mantegazza P.DIAPM-RTAI:why's, what's and how's. Proceedings of the Real Time Linux Workshop. Vienna, Austria, 1999.
[6] Montavista, www.mvista.com
[7] Morgan, K. in Linux Devices, www.linuxdevices.org
[8] Real Time Linux website, www.realtimelinux.org
[9] RTLinux, www.rtlinux.org