Mechatronics Real Time Linux
A RTOS Live CD Based on Linux

Rafael Aroca[1], Dalton Tavares[1], Glauco Caurin[1]

[1] Escola de Engenharia de São Carlos (EESC)
Departamento de Engenharia Mecânica
Laboratório de Mecatrônica
Universidade de São Paulo (USP) – São Carlos, SP – Brazil

rafaelaroca@ieee.org, {dmatsuo,gcaurin}@sc.usp.br

Abstract. *Real Time Operating Systems (RTOS) play a big role when computers are used to control mechanical and electrical devices that need reliable and deterministic control. Although several commercial RTOSes with easy to use interfaces and integrated development environments are available in the market, Linux is being gradually more used because of its low cost and free software philosophy. Potential users, which are not familiarized with Linux, may have trouble setting up a Real Time Linux based system. In this paper we describe the results obtained by tailoring an easy to use Real Time Linux Live CD and the feedback obtained by the realization of mechatronics experiments on undergraduate courses.*

1. Introduction

Real Time Systems are being developed since 1940 [Laplante 2004]. From military and aerospace applications using hard Real Time systems, to ordinary DVD Players running soft Real Time systems in every house. Building a Real Time System is not a simple task, and most commercial solutions use expensive proprietary tools due to strongly reliable and deterministic constraints needed in such systems. Thanks to the free software initiatives, many open source tools have been successfully adopted in recent years for mission critical situations [Irwin et al. 2002].

Many efforts were made to transform the Linux Kernel into a Real Time Kernel. As a result, several flavors of Real Time Linux arose - e.g. Xenomai [Xenomai 2007], RTLinux[1] [Yodaiken 1999], KURT [University of Kansas, Center for Research, Inc. 2007] and RTAI [Dipartimento di Ingegneria Aerospaziale – Politecnico di Milano 2007], which are free implementations. There are also proprietary solutions like Montavista [Montavista 2007], BlueCat Linux [LynuxWorks 2007] and WindLinux[2] [Wind River 2007b]. Due to the growing interest in Real Time Linux by the industry, RTLinux was acquired by Wind River in February of 2007 [Wind River 2007a].

A Real Time System is focused on determinism [Aarno 2007, Laplante 2004]. It may not be fast at all, but it must not break deadlines in any way. If a task must be

[1]RTLinux is the name of a commercial version of Real Time Linux. The company that provides RTLinux also offers a free version of it with fewer resources and some restrictions. Although RTLinux appears to be a general term for any Real Time Linux, care should be taken not to confuse the term.
[2]Developed by the company of the renowned VxWorks RTOS, Wind River.

executed at each second, a delay on that execution could cause fatal errors resulting in material loss or even harm humans. Soft Real Time Systems do not cause any safety critical drawbacks if a task is delayed. For example, a computer running a video consists on a Soft Real Time situation. Hard Real Time, on the other hand, can cause damage if minimum delays happen, such as not stopping a robot that will collide against a human being.

Due to the demands on Real Time Systems, it is commonplace to use Real Time Operating Systems to support the development of a product or a project that have Real Time constraints, hence shortening the development times and reducing risks that were already considered by the RTOS provider. Although existent RTOSes are available as great tools, they need to be installed, configured and prepared for a given application. This task usually requires a specialist, making things harder for non experienced users. The difficulties are worse when we're talking about Linux and free software because there is not a company behind the product to give technical support and solve questions. The users have to ask the community, usually in mailing lists, which does not have a guaranteed response.

At first, our intended audience consists on undergraduate professors teaching Real Time systems. Considering the limited period of time of a regular class, which must include the preparation of lectures/experiments, presentation of the concepts and a practical experiment to simplify its understanding by the target audience. The environment used must be simple enough to configure and user friendly, otherwise it hardly will be adopted. Our proposal is the construction of a Live CD, which can be run in any computer without installation or pre-configuration. When the CD is inserted, the system boots up and offers a fully featured Real Time development platform. The existing Real Time Live CDs will be discussed, followed by the resources offered by the Mechatronics Real Time Linux (MecRTL). The main objective behind the creation of this Live CD distribution is to create an easy to use development environment to teach the inner workings of a Real Time system. Today this is accomplished by means of theoretical classes and proof of concept exercises of each given subject using MecRTL. Some examples will be presented throughout the text.

2. Live CDs

A Live CD allows any user to run different operating systems or applications without having to install it on the computer. The user just need to turn the computer on and boot it using the CD-ROM drive or a USB removable flash disk. The live system can be used as a fully featured operating system. After removing the media and rebooting the computer, the old operating system is restored without any changes. Some well-known examples of general purpose Live CDs include the Kurumin Linux [Guia do Hardware 2007], Knoppix Linux [Knopper 2007] and Slax[Matejicek 2007]. There is even a Windows XP Live CD available from Microsoft.

Initiatives focusing specifically Real Time applications include Real Time Linux Live CD [Palli 2007], Koan Software's Real Time Linux [KOAN software engineering 2007] and QNX Neutrino [QNX Software Systems 2007]. Most bootable RTOS Live CDs are also available for benchmark tests or to provide a small system image for embedded systems. A notorious Live CD is the RTAI Testsuite

Live CD [Issaris 2007] which uses only 8MB of the CD-ROM space and tests the Real Time characteristics of the booted computer. It sends the data later to a database over the Internet which gathers information from thousands of volunteers concerning jitter, latency and other Real Time parameters. The database can be consulted on-line in the CD-ROM website.

The next sections will discuss briefly the Real Time Live CDs mentioned before and its application scope (Section 2.1). It will also introduce the motivation for creating a new Live CD and will discuss how it was customized (Section 2.2).

2.1. Real Time Live CDs

Koan Software's Real Time Linux [KOAN software engineering 2007], called KaeilOS with about 20MB of space, was specially designed for embedded systems. Although the software maintainer describes KaeilOS as a GPL Open Source licensed and 'royalty free' system, currently it is no longer publicly available.

QNX Neutrino is a commercial Real Time Operating System that had been used in hundreds of Real Time projects along the past years[3]. QNX provides a free Live evaluation CD that boots the QNX Neutrino operating system and offers an Eclipse based development environment. It uses a microkernel operating system where every driver, application, protocol stack, and file system runs outside the kernel, in the safety of memory-protected user space. As a result, virtually any component can fail — and be automatically restarted — without affecting other components or the kernel.

Considering the cost involved on the acquisition of the described systems and the steep learning curve to create, for example, the experiment basis for a Real Time undergraduate class, we consider a strategical impossibility to use KaeilOS or QNX Neutrino. The Real Time Linux Live CD, developed by Gianluca Palli, a PhD student from University of Bologna, Italy, might be the closer example to the approach we chose. The CD includes RTAI 3.4 and Scilab 4.0, which is an open source alternative to Matlab. The CD also contains Real Time examples and a development environment [Palli 2007]. Considering the professor point of view this environment would be enough, although it requires a basic understanding of the Linux compilation and linking system (i.e. basic uses of gcc, ld and make files). Another minor drawback is the fact that this live CD does not have a centralized way to inspect student practices per experiment. Depending on the way the the experiment results are collected, the student evaluation can be a reasonably time consuming task (e.g. by collecting them manually via e-mail or a USB disk after classes).

2.2. Mechatronics Real Time Linux – MecRTL

Before introducing the modifications implemented on Slax to compose what we call Mechatronics Real Time Linux or MecRTL, it is necessary to describe the general purpose distribution that composes its basis: Slax. Slax [Matejicek 2007] is a general purpose Live CD based on Slackware Linux, one of the oldest Linux distributions. It is a modular distribution, which can be easily modified according to each situation needs. It also has two powerful network features. The first is called websave, and allows any user to type the command "websave", and save its own home directory in a web server. The user can restore his home directory in any machine afterwards, simply booting Slax with the

[3]More information can be found at http://www.qnx.com/markets/index.html

command "webrestore". The other feature is the ssh file system (sshfs), which is possible thanks to Linux support for user space file systems. A Slax user simply needs to run "sshfs user@server /mountpoint" and it will have a remote directory mounted using the well known and used ssh protocol.

Changes to the CD-ROM file system can be easily done by creating files in the CD-ROM's $rootcopy$ directory. Every file and directory that is in this directory will be copied into the root file system after the boot. If the file already exists it will be replaced by the new one inserted at the rootcopy. Users can also make their own Slax packages that have to be copied into the modules directory, so that they are automatically executed at each boot. Section 2.2.1 describes the Real Time patch we use to modify the Linux Kernel task scheduling mechanism, so we can turn a non-real time system on a real time system.

2.2.1. Real Time Application Interface (RTAI)

Traditional Real Time Operating Systems (RTOSes) are developed from scratch following rigorous requisites and concerns, to provide maximum reliability, determinism and stability. Converting a preemptive kernel originally designed for servers and desktops into a deterministic Real Time one is the main contribution of RTAI [Lennon 2001].

One interesting approach taken by the RTAI team uses the solution based on the nanokernel architecture proposed by the Adaptive Domain Environment for Operating Systems (ADEOS) Project [Dozio and Mantegazza 2007, Yaghmour 2007]. The nanokernel is added to the Linux Kernel as a patch that intercepts interrupts and hardware access requests from the original Linux Kernel. The RTAI nanokernel executes all Real Time Tasks and schedules all the rest of the Linux operating system when there is idle Time from the Real Time execution.

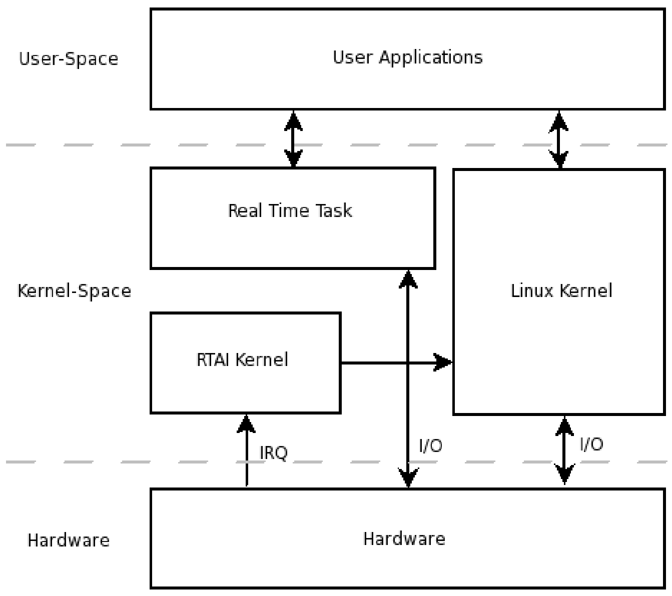


Figure 1. RTAI Architecture

RTAI provides the required schedulers along with a wealth of services. The RTAI schedulers can be scheduled directly from within interrupt handlers so that Linux has no chance of delaying any RTAI hard Real Time activity. It is important to observe the fact that RTAI schedulers and related services allow to symmetrically work both in user and kernel space, by using the same APIs everywhere. Any communication and interaction between the user space and kernel space is simple in RTAI, down to full hard Real Time interrupt handling in user space. It is possible to implement hard Real Time multitasking applications in kernel space also, by either using standard kernel threads or RTAI proper tasks. The possibility of embedding a control system as part of the kernel allows achieving maximum execution efficiency.

Usually, a Real Time task consists of a task running inside of a kernel module, but there is also a RTAI extension called LinuX Real Time (LXRT) which allows Real Time programs to be written and run in user space. Figure 1 shows RTAI interaction with the rest of the Linux system. All interrupts are captured by the RTAI Kernel, and forwarded to the original Linux Kernel if the received interrupt is not relative to a Real Time task.

2.2.2. Slax + RTAI = MecRTL

Mechatronics Real Time Linux (MecRTL) take advantage of the features of both RTAI and Slax, adding Slax modularity and simplicity with the Real Time features of RTAI. The bootable CD also contains a KDevelop Integrated Development Environment and a KDE graphical user interface with an office suite and an Internet browser. The network interface cards are automatically detected at boot Time and configured.

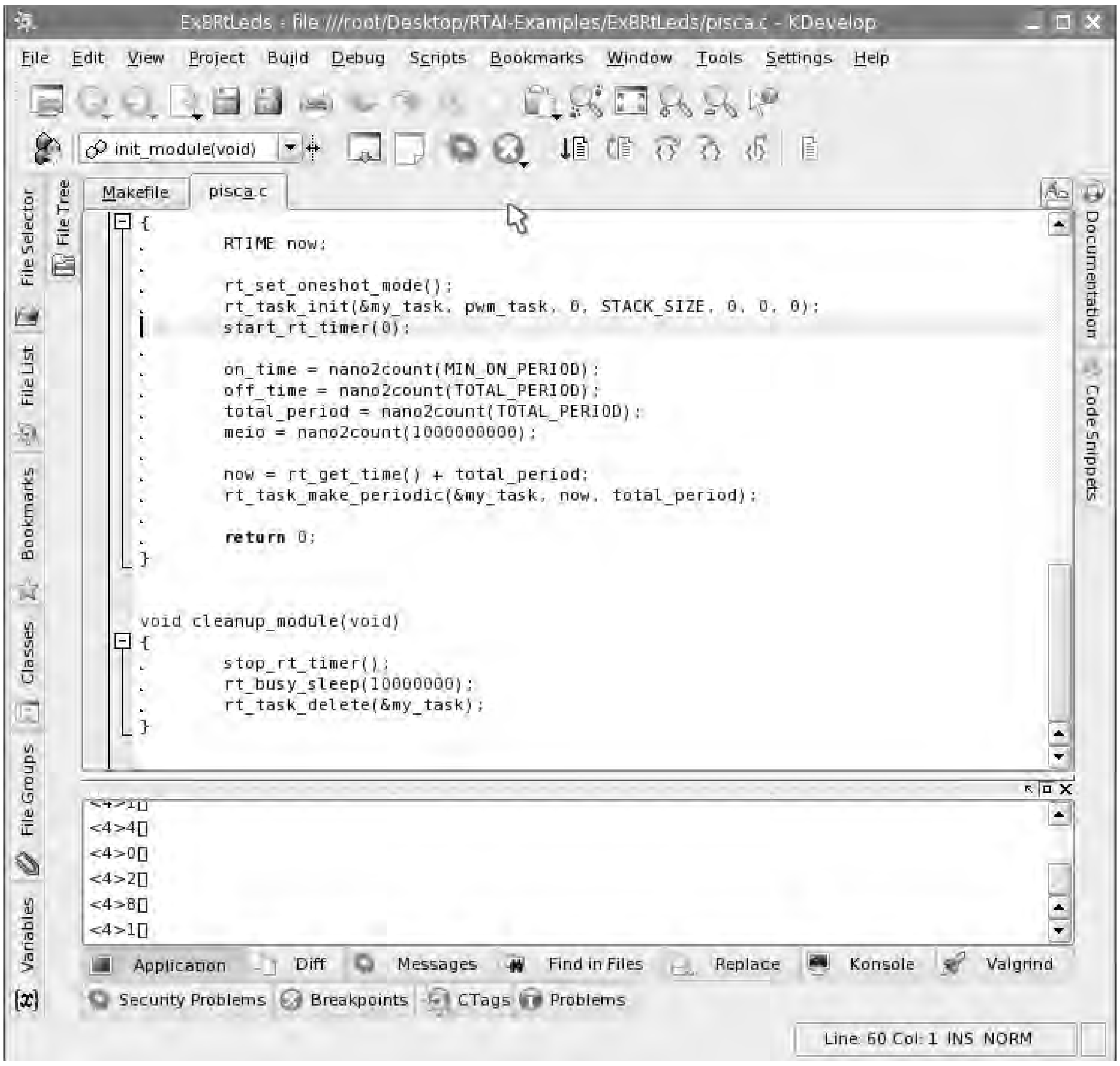
The NIST Real Time Tutorial [NIST 2007] was also added to the CD, with hands on step by step examples to learn Real Time Linux with RTAI. Some examples developed in the laboratory are also available ranging from a ‘Hello World’ in Real Time to pulse width modulation (PWM) test applications. The examples that use hardware I/Os are based on simple parallel port circuits, allowing anyone to run the examples on PCs (considering most PCs have parallel ports). Figure 2 shows the KDevelop environment running an example that blinks a LED. The kernel output can be seen in the messages tab of KDevelop.

3. Using the CD in an undergraduate course

One of the purposes intended for this CD is to give Real Time classes to undergraduate students. Most of them had never used Linux and were afraid of not knowing how to use it. The approach chosen for these classes include the development of experiments with MecRTL, each of them with an increasing level of complexity. CDs were distributed to the students and all of them were able to develop their first Real Time programs within two classes. For some students it took only a few explanations to create their first programs in less than one hour. The CD includes a Real Time Kernel based on RTAI, the KDE graphical user interface and a graphical integrated development environment (IDE). The CD size is about 360MB, with plenty of space for customizations and new additions.

3.1. Creating a Real Time Application

This section presents the specifics for building a Real Time Application using the proposed Live CD. As the Real Time tasks are kernel modules and the idea is to offer an



```
{
        RTIME now;

        rt_set_oneshot_mode();
        rt_task_init(&my_task, pwm_task, 0, STACK_SIZE, 0, 0, 0);
        start_rt_timer(0);

        on_time = nano2count(MIN_ON_PERIOD);
        off_time = nano2count(TOTAL_PERIOD);
        total_period = nano2count(TOTAL_PERIOD);
        meio = nano2count(1000000000);

        now = rt_get_time() + total_period;
        rt_task_make_periodic(&my_task, now, total_period);

        return 0;
}


void cleanup_module(void)
{
        stop_rt_timer();
        rt_busy_sleep(10000000);
        rt_task_delete(&my_task);
}
```

```
<4>1▯
<4>4▯
<4>0▯
<4>2▯
<4>8▯
<4>1▯
```

Figure 2. Mechatronics Real Time Linux Screenshot

easy to use interface, some scripts were built to insert and remove Linux Kernel Modules (LKMs) in the running kernel transparent to the user. Therefore, instead of running *insmod module.ko* to start the task and *rmmod module.ko* to stop it, a user simply needs to click on KDevelop run/stop icon.

That was achieved using a shell script with signal handlers, as the KDevelop start button runs a program, and Stop button sends a TERMINATE signal. Listing 1 demonstrates how that can be accomplished using a shell script. The shell script simply starts the desired LKM with insmod and declares a signal handler for the terminate signal. When the signal is received, the shell script removes the LKM with rmmod and then exits.

4. Experimental: controlling a Scara robot

One of the applications that were built on top of the MecRTL was a control system for an IBM Selective Compliant Articulated Robot Arm (SCARA) model 7545 (Figure 3) industrial robot with four axis. The SCARA robot have been originally manufactured by IBM in 1980 and the electronics and software systems had to be redesigned and rebuilt by the mechatronics laboratory team.

Listing 1. Wrapper written in Unix shell script to run and stop the tasks.

```
#!/bin/sh
MODULE="test1"
#Trap Handler - see below
tr_handler() {
        rmmod $MODULE
        echo "Ended $MODULE"
        exit 0
}
#Handles signals 1 (HANGUP) 2 (INTERRUPT) and 15 (TERMINATE)
trap "tr_handler" 1 2 15

#Loads the LKM
echo "Loading $MODULE"
insmod $MODULE.ko

#Shows the kernel output in KDevelop messages window
tail -f /var/log/messages
```

The RTOS adopted for Real Time control is the MecRTL. One of the most critical tasks of the MecRTL system is to run the Proportional/Integral/Derivative (PID) control loops. A PID controller is a closed loop system that checks the current joint position of the robot, and set the motor voltage levels to maintain the robot on a certain position. This loop is executed in one millisecond cycles and if we try to move (push or pull) the robot arm, it returns it to its original position. To move the robot, the PID set point has to be changed. Each robot axis has a motor which is controlled by its own PID task inside the MecRTL, summing 4 Real Time tasks.

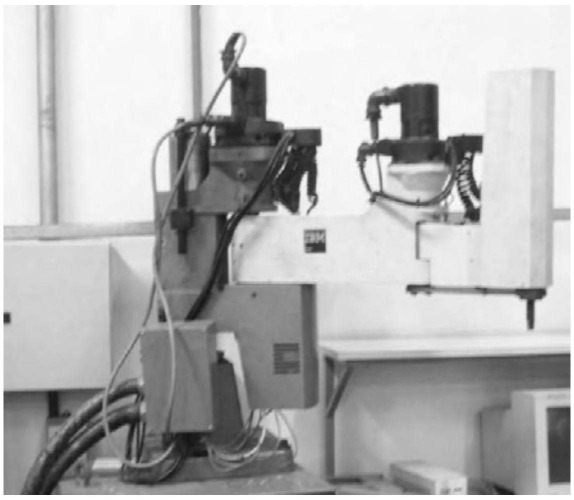


Figure 3. SCARA Robot

5. Availability

MecRTL is a free software project under the terms of the General Public License (GPL). For further information and download, refer to the website `http://www.mecatronica.eesc.usp.br/~aroca/slax-rt/`.

6. Conclusion and Future work

In this paper we presented an easy to use Real Time Linux Live CD developed at the Mechatronics Laboratory, which aims to allow developers to deal only with their Real Time needs. Different from traditional RTOSes, the Mechatronics Live CD does not need a complex setup or configuration. The Live CD is available as a free software and can be downloaded and burned into a CD to be used. The CD is currently being used to teach Real Time courses and to control a retrofitted IBM 7545 SCARA robot (which is also being used as part of the course). For now, the user must be aware of all the complexities of the system, meaning he/she must understand user/kernel space concepts, and develop the program for kernel space. In the near future, the objective is the development of user space libraries using LinuX Real Time (LXRT) in order to standardize the creation of user space programs. In this case, a LXRT kernel module takes care of the transposition of a command from user space to kernel space simplifying the user interaction with the system.

7. Acknowledgments

We would like to thank the Slax and RTAI developers, specially Tomas Matejicek and Paolo Mantegazza, for creating these softwares in which we based our Live CD.

References

Aarno, D. (Last Access on Mar 2007). Evaluation of real-time linux derivatives for use in robotic control. On-Line: http://www.nada.kth.se/ bishop/resources/rtos.pdf.

Dipartimento di Ingegneria Aerospaziale – Politecnico di Milano (Last Access on Mar 2007). Rtai 3.4 user manual rev 0.3. On-line: https://www.rtai.org/.

Dozio, L. and Mantegazza, P. (Last Access on Mar 2007). Linux real time application interface (rtai) in low cost high performance motion control. On-line: http://www.aero.polimi.it/ rtai/documentation/conferences/motioncontrol2003.pdf.

Guia do Hardware (Last Access on Mar 2007). Kurumin linux. On-line: http://www.guiadohardware.net/gdhpress/kurumin/.

Irwin, P., Richard, L., and Johnson, J. (2002). Real-time control using open source rtos. In Lewis, H., editor, *Advanced Telescope and Instrumentation Control Software II*, volume 4848, pages 560–567. SPIE.

Issaris, P. (Last Access on Mar 2007). Rtai testsuite livecd – collecting hardware's hard real-time performance data. On-line: http://issaris.homelinux.org/ takis/projects/rtai/livecd/.

Knopper, K. (Last Access on Mar 2007). What is knoppix? On-line: http://www.knopper.net/knoppix/index-en.html.

KOAN software engineering (Last Access on Mar 2007). Koan embedded software engineering. On-line: http://www.koansoftware.com/.

Laplante, P. A. (2004). *Real-Time Systems Design and Analysis*. Willey Interscience.

Lennon, A. (May 2001). Embedding linux. *IEE Review*, 47(3):33 – 37.

LynuxWorks (Last Access on Mar 2007). Embedded linux: Bluecat linux - robust embedded-linux operating system based on linux 2.6 kernel. On-Line: http://www.lynuxworks.com/embedded-linux/embedded-linux.php.

Matejicek, T. (Last Access on Mar 2007). - slax - your pocket os. On-line: http://www.slax.org/?lang=en.

Montavista (Last Access on Mar 2007). Real-time linux. On-line: http://www.mvista.com/products/realtime.html.

NIST (Last Access on Mar 2007). Real-time linux. On-Line: http://www.isd.mel.nist.gov/projects/rtlinux/.

Palli, G. (Last Access on Mar 2007). Rtai-knoppix (md5sum) - knoppix based rtai livecd. On-line: http://www-lar.deis.unibo.it/people/gpalli/#software.

QNX Software Systems (Last Access on Mar 2007). Qnx neutrino realtime operating system. On-line: http://www.qnx.com/products/rtos/.

University of Kansas, Center for Research, Inc. (Last Access on Mar 2007). Kurt-linux user manual - draft. On-line: http://www.ittc.ku.edu/kurt/papers/user-manual-DRAFT.pdf.

Wind River (Last Access on Mar 2007a). Wind river acquires hard real-time linux technology from fsmlabs. On-line: http://www.windriver.com/news/press/pr.html?ID=4261.

Wind River (Last Access on Mar 2007b). Wind river linux center. On-line: http://www.windriver.com/linux/.

Xenomai (Last Access on Mar 2007). Xenomai: Real-time framework for linux. On-line: http://www.xenomai.org.

Yaghmour, K. (Last Access on Mar 2007). Adaptative domain environment for operating systems. On-Line: http://www.opersys.com/adeos/.

Yodaiken, V. (1999). The RTLinux manifesto. In *Proceedings of The 5th Linux Expo, Raleigh, NC*.