UNIVERSIDAD CATÓLICA NUESTRA SEÑORA DE LA ASUNCIÓN.

RTAI – Real Time Application Interface.

Francisco Resquín Acosta.
Matricula: 52065

Teoría y Aplicación de la Informática 2– T.A.I 2
07/09/2009

Índice.

1- Introducción.

Linux es un sistema multitareas. La desventaja de Linux es que no puede garantizar la respuesta en el tiempo de los procesos. Sin embargo hay áreas, donde las aplicaciones requieren respuesta en tiempo real, como dispositivos robóticos, computadoras usadas en el control de la salud y en aplicaciones militares, y varios sistemas integrados que usan diferentes tipos de dispositivos.

Existen varios sistemas operativos de tiempo real comerciales disponibles, como QNX, AMX, RTKernel, los cuales son más pequeños y convienen generalmente ser usados en sistemas integrados. Sin embargo, algunos desarrolladores y investigadores han señalado que si bien hay necesidades para un núcleo de tiempo real, sería útil disponer de varios drivers de dispositivos, protocolos de redes, y otras características ofrecidas y disponibles en Linux.

Por las necesidades dichas anteriormente, varias implementaciones de Linux en tiempo real han sido presentadas. Sin embargo la mejor solución es RTlinux, la cual ha sido usada en forma básica como una solución de Linux en tiempo real. RTlinux inserta un pequeño kernel de tiempo real abajo del estándar kernel de Linux y trata al kernel de Linux como un proceso de tiempo real. La desventaja de este enfoque es que las tareas de tiempo real operan en un espacio de kernel y en el caso de un bug del software podría causar daños considerables.

Una solución basada en RTlinux es Real time Application Interface (RTAI) desarrollado en la politécnica de Milán, en el departamento de ingeniería Aeroespacial. De las soluciones de Linux de tiempo real disponibles, es la más activamente actualizada y mantenida. Las características más importantes de RTAI incluye varios métodos de comunicación flexible entre procesos, y un API simétrico permitiendo la creación de de tareas en tiempo real en el espacio de usuario, evitando la desventaja de operación en el espacio de kernel.

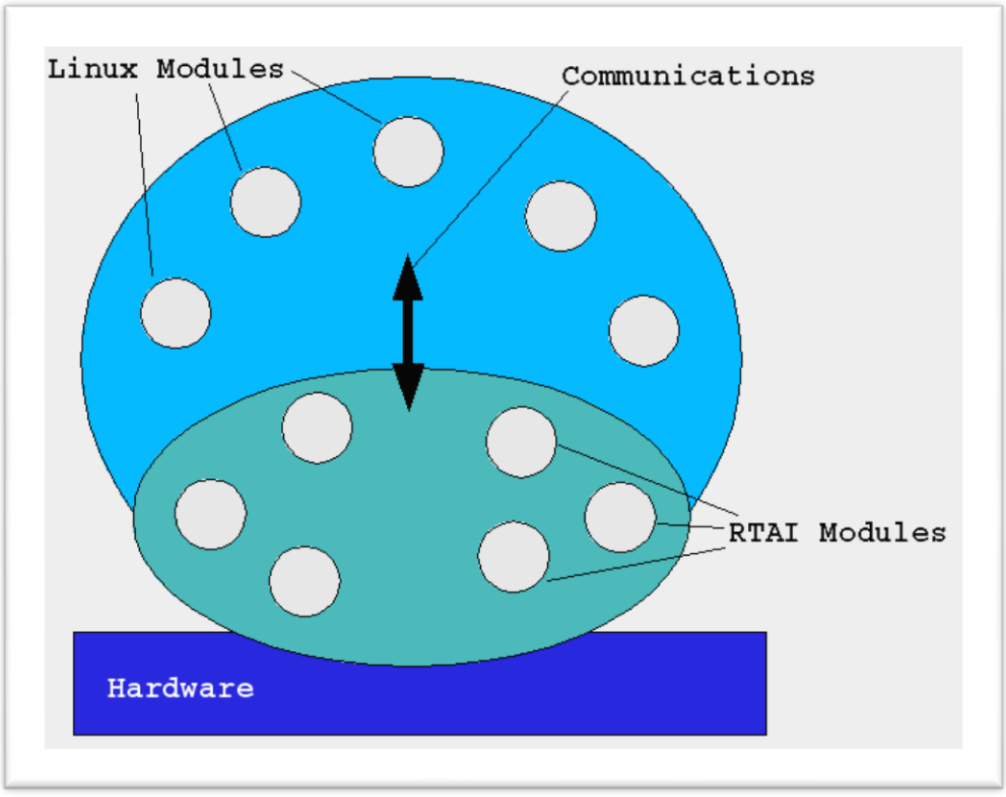
2- Qué es el RTAI?

RTAI es una implementación de Linux para tiempo real basada en RTLinux. Añade un pequeño kernel de tiempo real bajo el kernel estándar de Linux y trata al kernel de Linux como una tarea con la menor prioridad. RTAI además proporciona una amplia selección de mecanismos de comunicación entre procesos y otros servicios de tiempo real.

RTAI provee garantía de una programación de tiempo real, pero conserva todas las características y servicios del Linux estándar. Adicionalmente, RTAI proporciona un módulo llamado LXRT para facilitar el desarrollo de aplicaciones de tiempo real en el espacio de usuario.

Licencia: GNU/GPL.

Esquema general del Sistema operativo Linux con el RTAI.



3- Fundamentos del RTAI.

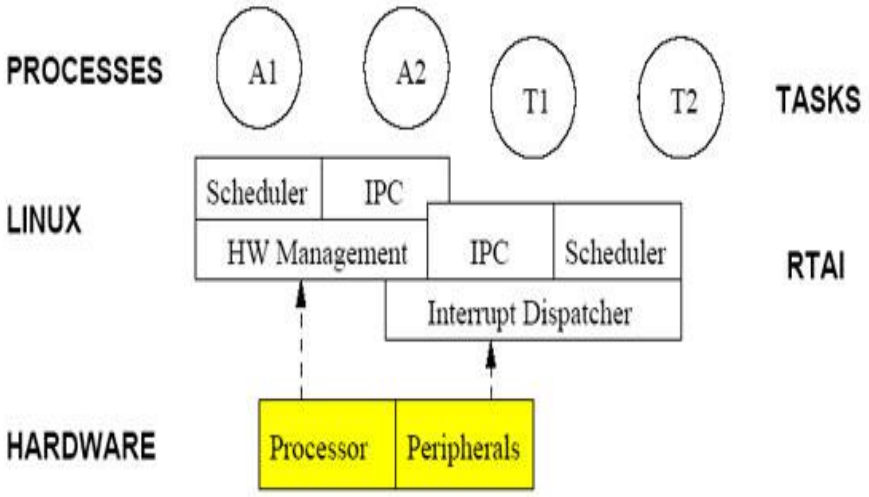
RTAI tiene una arquitectura similar a RTLinux. Al igual que RTLinux, RTAI trata el kernel estándar de Linux como una tarea de tiempo real con la menor prioridad, lo

que hace posible que se ejecute cuando no haya ninguna tarea con mayor prioridad ejecutándose. Las operaciones básicas de las tareas de tiempo real son implementadas como módulos del kernel al igual que RTLinux. RTAI maneja las interrupciones de periféricos y son atendidas por el kernel de Linux después de las posibles acciones de tiempo real que hayan podido ser lanzadas por efecto de la interrupción.

Así, Linux no sufre cambios en su operación desde el punto de vista del usuario o el kernel de Linux, salvo que esté autorizada a ejecutar sólo cuando no hay tareas en tiempo real de ejecución.

La siguiente figura muestra la arquitectura básica de RTAI, que es muy similar a la de RTLinux. Las interrupciones se originan en el procesador y en los periféricos. Las originadas en el procesador (principalmente señales de error como división por cero), son manejadas por el kernel estándar, pero las interrupciones de los periféricos (como los relojes) son manejadas por RTAI Interrupt Dispatcher. RTAI envía las interrupciones a los manejadores del kernel estándar de linux cuando no hay tareas de tiempo real activas. Las instrucciones de activar/desactivar las interrupciones del kernel estándar son reemplazadas por macros que se enlazan con las instrucciones de RTAI. Cuando las interrupciones están desactivadas en el kernel estándar, RTAI encola las interrupciones para ser repartidas después de que el kernel estándar haya activado las interrupciones de nuevo.

Adicionalmente, se puede ver en la figura el mecanismo de comunicación entre procesos (IPC), que esta implementado de forma separado por Linux y por RTAI. También existe un planificador (sheduler) distinto para Linux y para RTAI.



4- HAL - Hardware Abstraction Layer.

Los desarrolladores de RTAI introducen el concepto de Real Time Hardware Abstraction Layer (RTHAL) que es usado para interceptar las interrupciones hardware y procesarlas después. RTHAL es una estructura instalada en el kernel de Linux que reúne los punteros a los datos internos del hardware relacionados en el kernel y las funciones necesarias por RTAI para operar. El objetivo de RTHAL es minimizar el número de cambios necesarios sobre el código del kernel y por tanto mejorar el mantenimiento de RTAI y del código del kernel de Linux. Con RTHAL, las diferentes operaciones (ej. manejar interrupciones) son más fáciles de cambiar y modificar sin tener que interferir con la implementación de Linux. Por ejemplo, la estructura de RTHAL contiene la tabla de manejadores de interrupción, la cual es una lista de las funciones que son llamadas para manejar las diferentes interrupciones. El cambio consiste únicamente en modificar unas 20 líneas del código del kernel de Linux y añadir unas 50 nuevas líneas.

Cuando RTHAL es instalado en Linux, las funciones y las estructuras de datos relacionada con la interacción con el hardware son reemplazadas por punteros a la estructura de RTHAL.

5- Planificación.

La unidad de planificación de RTAI es la tarea. Siempre hay al menos una tarea, llamada kernel de Linux, que ejecuta como la tarea de menor prioridad. Cuando las

tareas de tiempo real son añadidas, el planificador da entonces mayor prioridad a éstas sobre la tarea del kernel de Linux. El planificador proporciona servicios tales como suspend, resume, yield, make periodic, wait until, que son usadas en varios sistemas operativos de tiempo real.

El planificador es implementado como un módulo del kernel dedicado (contrario a RTLinux) lo que facilita la implementación de planificadores alternativos si es necesario. Actualmente hay tres tipos de planificadores dependiendo del tipo de máquina:

- Uniprocesador (UP).

El planificador Uniproceso realiza un algoritmo de programación para seleccionar la tarea que se va a ejecutar en un CPU. Como tal su funcionamiento es muy sencillo: cualquier proceso tiene una prioridad alta de tomar la CPU. En efecto es una lista de múltiple prioridades programada con soporte para herencia prioritaria. En este esquema Linux es una tarea de tiempo real, como cualquier otro pero manteniendo el nivel de menor prioridad.

- Multiprocesador simétrico (SMP).

Está diseñado para maquinas SMP y proporciona un interfaz para las aplicaciones de forma que es posible seleccionar el procesador y procesadores que deben ejecutar una tarea. Si el usuario no especifica un procesador para la tarea, SMP selecciona el procesador en función de la carga de trabajo.

- Multi-Uniprocesador (MUP)

Puede ser usado con ambos, pero al contrario que SMP, a las tareas se les debe especificar el procesador que deben usar. Viéndolo por el lado positivo, el planificador MUP permite unos mecanismos de tiempo más flexibles para las tareas que los planificadores SMP y UP.

6- Características.

Con el objetivo de hacer el desarrollo de aplicaciones más fáciles y flexibles posibles, los desarrolladores de RTAI han introducido diferentes mecanismos para la comunicación entre procesos (IPC), entre las tareas de tiempo real y los procesos

en el espacio de usuarios. Como añadido al mecanismo IPC, RTAI proporciona servicios de manejo de memoria y threads compatibles con Posix.

6.1- Comunicación entre procesos (IPC - Inter-process comunication).

RTAI proporciona una variedad de mecanismos para la comunicación entre procesos. Aunque los sistemas Unix proporcionan mecanismos similares a IPC para los procesos en el espacio de usuario, RTAI necesita proporcionar una implementación propia para que las tareas de tiempo real puedan usar este mecanismo y no usen el estándar del kernel de Linux. Los diferentes mecanismos de IPC están incluidos como módulos de kernel, lo que facilita la carga cuando son necesarios. Como ventaja adicional el uso de módulos para los servicios, IPC facilita el mantenimiento y la expansión.

El antiguo mecanismo básico de comunicación de RTAI eran los FIFOs. FIFO es un asíncrono y no bloqueante canal de comunicación entre los procesos de Linux y las tareas de tiempo real. La implementación de RTAI de FIFO está basada en la implementación de RTLinux, pero RTAI proporciona algunas características que no son posibles en RTLinux. Primeramente RTAI puede lanzar señales cuando hay eventos en el FIFO (escritura de nuevos datos). Los procesos en el espacio de usuario pueden entonces crear un manejador para la señal por los mecanismos estándar de Unix. Sin embargo, este mecanismo no es necesario para los procesos de usuario que quieran leer y escribir del FIFO. Adicionalmente, puede haber múltiples lectores y escritores en el FIFO, cosa que no es posible en la versión de RTLinux. Finalmente, los identificadores FIFO pueden ser dinámicamente localizados con un nombre simbólico. Antes era necesario establecer un identificador global para el FIFO, lo que causaba problemas cuando múltiples e independientes procesos y tareas lo usaban.

Los semáforos son otra herramienta básica de sincronización entre procesos usada en los sistemas operativos. RTAI proporciona un API para usar semáforos, aunque cada semáforo está técnicamente asociado a un FIFO, por tanto cada semáforo usa una entrada global del FIFO. Como añadido al servicio básico de semáforos, un semáforo puede estar asociado con un reloj, el cual puede ser usado para despertar un proceso encolado en un semáforo, incluso cuando el semáforo aun esté cerrado.

La memoria compartida proporciona una alternativa a IPC y al paradigma FIFO cuando un modelo de comunicación diferente es requerido. La memoria compartida es un bloque común de memoria que puede ser leído y escrito por un proceso y una tarea en el sistema. Como los diferentes procesos pueden operar de forma asíncrona en la región de memoria, es necesario un diseño para asegurar que los datos no sean sobre escritos de forma intencionada.

Finalmente, el método más flexible de IPC quizá sean los mailboxes. Cualquier número de procesos pueden enviar y recibir mensaje de y desde un mailbox. Un mailbox almacena mensajes hasta un límite que se defina, y contrario a los FIFOs, mailbox preserva los mensajes que están en el límite. Puede haber un número arbitrario de mailbox activos en el sistema simultáneamente. RTAI también facilita la comunicación entre procesos mediante RPC.

6.2 - Gestión de memoria.

En las primeras versiones de RTAI la memoria tenía que ser asignada estáticamente y no era posible la asignación en tiempo real. Sin embargo en las actuales versiones se incluye un módulo gestor de memoria que permite la asignación dinámica de memoria por parte de las tareas de tiempo real usando un interfaz basado en una librería estándar de C.

RTAI pre asigna trozos de memoria antes de la ejecución de tiempo real. Cuando la tarea de tiempo real llama a la función rt_malloc(), la respuesta que obtiene es el trozo pre asignado. Antes de que el espacio se agote, RTAI reserva nuevos trozos de memoria (pre asigna) para futuras llamadas. De manera similar ocurre con la función rt_free(), en este caso, se libera la memoria pre asignada a la espera de futuras reservas. Cuando la suma de memoria liberada es mayor que un valor para una marca, se ordena su liberación.

6.3 - Threads Posix.

RTAI tiene módulos que proporcionan la implementación de threads de acuerdo al estándar POSIX 1003.1c. Usando las operaciones especificadas en el estándar, el usuario puede manejar de manera similar a como lo hace con los threads posix convencionales, excepto en cuanto a los conceptos de joining y detaching.

Los threads de un mismo proceso comparten el espacio de memoria, por tanto es fácil el intercambio de información entre ellos, sin embargo, el uso de áreas

memoria compartida son necesarias para la sincronización. También se proporcionan el mecanismo de mutex y de variables de condición.

6.4 - LXRT: User-space Interface to RTAI.

LXRT es un API para RTAI que hace posible el desarrollo de aplicaciones de tiempo real en el espacio de usuario sin tener que crear módulos para el kernel. Esto es útil en primer lugar porque el espacio de memoria destinado al kernel no está protegido de accesos inválidos, lo que puede provocar la corrupción de datos y el mal funcionamiento del kernel de Linux. En segundo lugar, si el kernel es actualizado, los módulos necesitan ser recompilados lo que puede provocar que sean incompatibles con la nueva versión.

Mientras se desarrolla una aplicación de tiempo real en el espacio de usuario el programador puede usar las herramientas estándar de depuración hasta que ya no contienen errores, pero en este caso se trata de procesos de tiempo real flexibles (soft). Además, los servicios proporcionados por las llamadas al sistema de Linux están disponibles para las tareas. La ventaja de esto es que cuando la aplicación está libre de errores puede convertirse a un módulo en el espacio del kernel, por lo que ya tendremos una tarea de tiempo real estricto. Sin embargo, al hacer esto, las llamadas al sistema utilizadas no sirven y deberán ser cambiadas por las proporcionadas por RTAI.

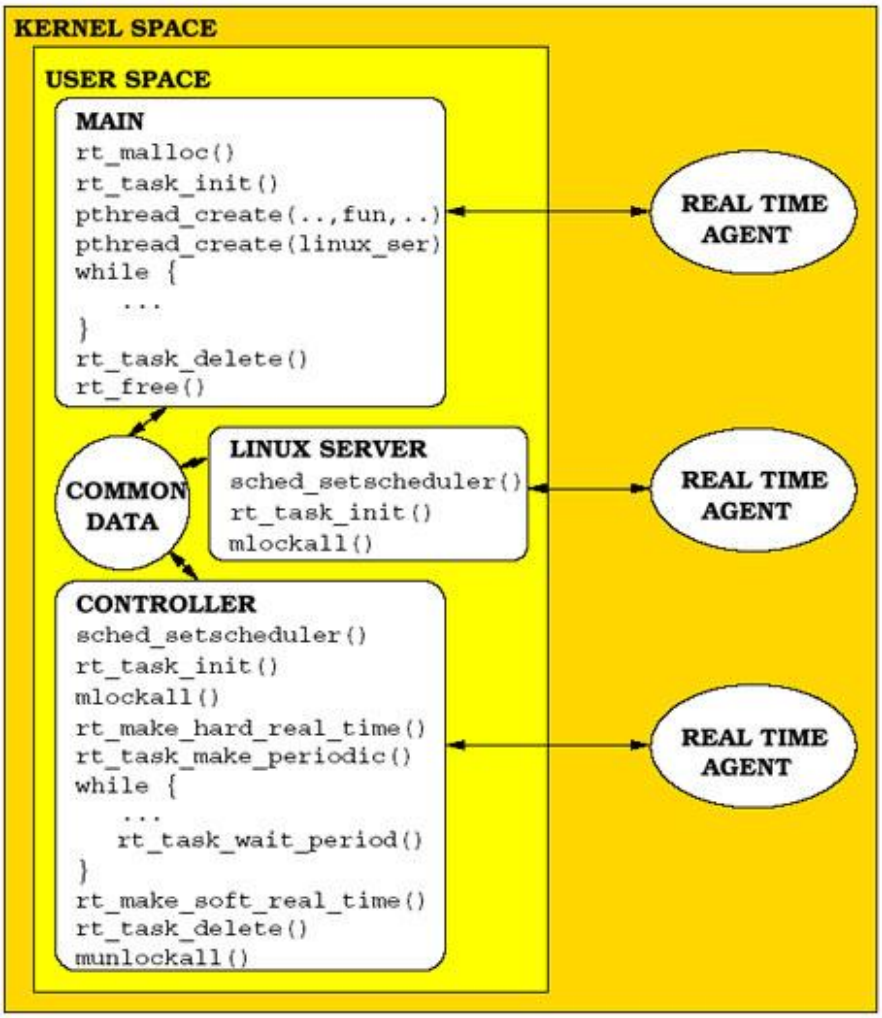
El cambio de la aplicación de procesos del espacio de usuario a tareas de tiempo real es fácil porque LXRT proporciona un API simétrico para la comunicación entre procesos y otros servicios de RTAI, lo que significa que el mismo API puede ser usado por las tareas de tiempo real y por los procesos del espacio de usuario. El mismo API de LXRT puede ser también usado cuando 2 procesos del espacio de usuario y 2 tareas de tiempo real se comunican entre sí, esto implica que varios relojes y mensajes del sistema proporcionados por LXRT puedan ser usados incluso cuando la aplicación no requiera tiempo real.

LXRT permite a las aplicaciones el intercambio dinámico entre tiempo real flexible y estricto mediante el uso de una simple llamada en el espacio de usuario. Cuando una aplicación esta en el modo de tiempo real flexible, usa el planificador de Linux, pero requiere el uso de la política de planificación FIFO. Sin embargo la planificación de procesos FIFO puede provocar la pérdida de ranuras de tiempo por

varias razones, por ejemplo, porque se está ejecutando un manejador de interrupciones y el planificador de tareas de RTAI.

Con el objetivo de facilitar el paso a tiempo real estricto, LXRT crea un agente en el espacio del kernel para cada proceso del espacio de usuario con requerimientos de tiempo real. Cuando el proceso entra en modo de tiempo real estricto, el agente desactiva las interrupciones, y mueve al proceso fuera del planificador de Linux y lo añade a la cola del planificador de RTAI. Ahora el proceso no es interrumpido por las interrupciones de otro proceso Linux. Para poder realizar este cambio el proceso debe asegurar que la memoria usada por el proceso este en la memoria RAM y debe desactivar la paginación usando la función mlockall(). Esta llamada no debe ser usada en modo tiempo real estricto.

Aunque los desarrolladores de RTAI ven bien el desarrollo de aplicaciones de tiempo real estricto usando LXRT, el tiempo de respuesta no es tan bueno como la ejecución de las tareas como módulos del kernel.



7- Aplicaciones.

Las aplicaciones en la actualidad son múltiples, en especial en el área del control, robótica y para sistemas militares, en sistemas que requieren toma de datos y procesamiento de estos en tiempo real. Este sistema de tiempo real fue diseñado ante la necesidad de contar con un sistema flexible y de gran capacidad para realizar control de sistemas, en tiempo real.

El RTAI, entonces como dijimos se utiliza ampliamente en todos los sistemas de control, por ejemplo, control de plantas, motores, y para el monitoreo de todo tipo de sistemas, ya sean médicos, de medición, y otros. En la robótica donde se debe tomar mediciones del medio, y procesar dicha información, y con tal información tomar decisiones en tiempo real. También es bastante utilizado en sistemas

militares donde se requiere gran capacidad de cómputo para tomar una decisión en tiempo real ante la toma de datos del exterior. En este momento otra área que está que utiliza con gran amplitud el RTAI son los wireless sensor network.

Con el gran crecimiento de los sistemas integrados y del control digital, podemos decir que en este momento casi todos los sistemas electrónicos de última tecnología cuentan u operan con este sistema, ya que gracias a su flexibilidad permite su implementación en varios procesadores. En la actualidad ya existen varios micros - procesadores que soportan este tipo de sistemas, permitiendo un gran desarrollo de sistemas electrónicos en general.

8- Desarrollos Futuros.

RTAI continúa creciendo y madurando a través de la contribución combinada de equipos de desarrolladores, quienes están animados por la arquitectura flexible, y alto performance. Mientras esta extensión de Linux en tiempo real es muy capaz, estable y maduro hoy en día, el trabajo está lejos de haber terminado. El futuro tiene muchas cosas, incluidas:

- Librerías de compatibilidad VxWorks.
- Librerías de compatibilidad pSOS.
- Herramientas de desarrollo mejorado.
- Herramientas de Debug mejorado.
- Funcionalidad de Ethernet en tiempo real.
- Sistemas de Archivo RAM de tiempo Real.
- Sistemas de archivos Flash – based.

Anexo A.

Diferencia entre RTAI y otros sistemas.

Tenemos que saber que el RTAI no es un sistema sino que es una función que nos permite Linux en realizar tareas en tiempo real.

Primero debemos saber que son sistemas operativos en tiempo real.

Un sistema operativo de tiempo real (RTOS -Real Time Operating System en inglés). Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible.

En la actualidad la utilización de esos sistemas operativos hace que el mercado y que mas empresas dedicadas a software creen dichos sistemas.

Sistemas de tiempo real de la empresa Red hat podemos citar entre sus principales.

Ecos: es un sistema operativo de tipo embebido que funciona sobre varias arquitecturas, entre ellas x86, PowerPC, MIPS o ARM.

Sobre su desarrollo se encuentra una potente empresa dedicada a Linux como lo es Red Hat. Es libre y gratuito.

Y no utiliza el kernel de linux, este sistema se ha desarrollado porque en muchas aplicaciones utilizar el kernel de linux no es la mejor opción.

Otros sistemas operativos de diferentes empresas.

QNX: es un sistema operativo de tiempo real basado en Unix que cumple con la norma POSIX. Es desarrollado principalmente para su uso en dispositivos empotrados. Desarrollado por QNX Software Systems empresa canadiense. Está disponible para las siguientes arquitecturas: x86, MIPS, PowerPC, SH4 (incluida la videoconsola Dreamcast con una versión muy limitada de este), ARM, StrongARM y xScale.

QNX está basado en una estructura de micronúcleo, que proporciona características de estabilidad avanzadas frente a fallos de dispositivos, aplicaciones, etc.

Photon o Photon microGUI es el sistema de ventanas (servidor y cliente) de QNX, aunque también funciona una versión X Window.

Los sistemas operativos de tiempo real son interesantes para situaciones donde sea absolutamente necesaria una toma continua de, por ejemplo, muestras de datos. Basándose en este interés, existen diversos proyectos para crear versiones en tiempo real de otros sistemas.

Está orientado a su utilización en microcontroladores y sistemas críticos.

VxWorks es un sistema operativo de tiempo real, basado en Unix, vendido y fabricado por Wind River Systems.

Como la mayoría de los sistemas operativos en tiempo real, vxWorks incluye kernel multitarea con planificador *preemptive* (los procesos pueden tomar la CPU arbitrariamente), respuesta rápida a las interrupciones, comunicación entre procesos, sincronización y sistema de archivos

Las características distintivas de VxWorks son:

- la compatibilidad POSIX
- el tratamiento de memoria
- las características de multi-procesador
- una shell de interfaz de usuario
- monitor de rendimiento y depuración de código fuente y simbólico.

VxWorks se usa generalmente en sistemas empotrados. Al contrario que en sistemas nativos como Unix, el desarrollo de VxWorks se realiza en un "host" que ejecuta Unix o Windows.

En la actualidad, vxWorks puede ejecutarse en practicamente todas las CPU modernas del mercado de sistemas empotrados.Esto incluye la familia de CPUs x86, MIPS, PowerPC, SH-4, ARM, StrongARM y xScale

Windows CE es el sistema operativo de Microsoft incrustado modular de tiempo real para dispositivos móviles de 32-bits inteligentes y conectados. Windows CE combina la compatibilidad y los ping a servicios de aplicación avanzados de Windows con soporte para múltiples arquitecturas de CPU y opciones incluidas de comunicación y redes para proporcionar una fundación abierta para crear una variedad de productos. Windows CE impulsa a los dispositivos electrónicos del cliente, terminales Web, dispositivos de acceso a Internet, controladores industriales especializados, computadoras de bolsillo, dispositivos de comunicación incrustados e incluso consolas de video juegos como fue en el caso de la Sega Dreamcast (1997 - 2001) con procesador SH4 de 128 Bits que ya con un sistema operativo propio, incluía compatibilidad con los kits para desarrollo de software de Windows CE.

Esta plataforma modular permite a los desarrolladores crear software para que la nueva generación de dispositivos móviles de 32-bits se integre con Windows e Internet.

Windows CE no es un subconjunto de Windows XP, o de Windows NT, sino que fue desarrollado a base de nuevas arquitecturas y una nueva plataforma de desarrollo. Aun así mantiene cierta conexión con sus hermanos. Windows CE tiene sus propias APIs para desarrollo, y necesita sus

propios drivers para el hardware con el cual va a interactuar. Windows CE no es un sinónimo de Windows XP en forma pequeña, incrustada o modular.

Windows CE también ha permitido la creación de un sistema denominado AutoPC , que consiste en un PC empotrado en un automóvil y que va ubicado donde normalmente va una radio. De esta manera permite controlar la radio, el reproductor de CD y revisar el correo electrónico. Windows CE también permite la creación de aplicaciones en tiempo real.

La última versión del Windows CE actualmente es Windows Mobile 6.1, Upgrade de Windows Mobile 6.0, sucesor de Windows Mobile 5.0, y sirve tanto para Pocket PC (PDA) como para SmartPhone.

Cabe destacar que este sistema operativo es el único producto de Microsoft que se distribuye junto con el código fuente, y usa una licencia llamada Shared Source, así pues permite al usuario final modificar el código fuente sin notificar al propietario.

Las diferencias fundamentales entre todos los sistemas descriptos anteriormente son las diferentes capacidades y los diferentes usos que se le pueden dar, cada sistema tiene su mejor desempeño en diferentes aplicaciones.

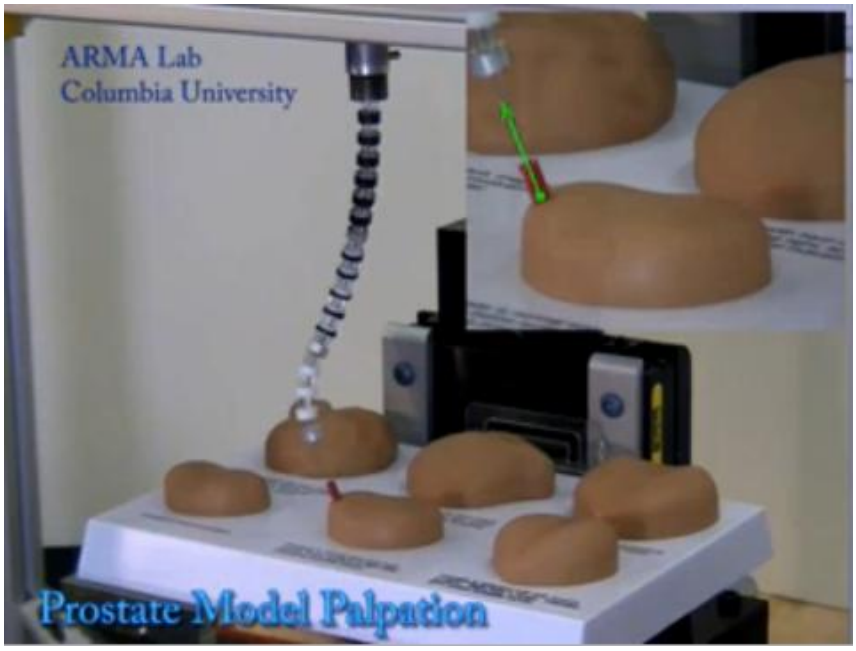
Pero la diferencia fundamental entre los sistemas citados y el RTAI es que son sistemas diseñados para tareas específicas, mientras que el RTAI se puede utilizar en diferentes aplicaciones. La desventaja que tiene el RTAI es que debemos tener una computadora o un sistema capaz de levantar alguna versión de Linux, lo cual hace que en diversas aplicaciones solo sea necesario otro sistema operativo para el tema de costos.

Anexo B.

Aplicaciones en Casos Reales en la Actualidad.

- Brazos Roboticos
- Controladores PID
- Palpador Robotico de Prostata

(Para examenes medicos)



- Controlador Digital de Velocidad de Motores en Industrias

Govt. Engineering College, Thrissur (Kerala, India)

9- Bibliografía.

1- Real time Application Interface. Pasi Sarolahti, Seminario de investigación en Real Time Linux. Universidad deHelsinky. Año 2002.

2- RTAI for Linux. Paolo Mantegazza. Departamento de Ingeniería Aeroespacial, Universidad de Milán.

3- Real Time Application Interface. Karim Yaghmour.

4- http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/RTAI-RealTime-Application-Interface/

5- https://www.rtai.org/documentation/

6- http://en.wikipedia.org/wiki/RTAI.