ANEXOS

SISTEMA DE MONITOREO EN TIEMPO REAL PARA PACIENTES MERIS

ANGELA MARÍA VARGAS ARCILA
CARLOS ALBERTO ORJUELA ZÚÑIGA

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Grupo de Ingeniería Telemática
Servicios Avanzados de Telecomunicaciones
Popayán
2009

ANEXO A

ANEXO A. INSTALACIÓN RTAI 3.7.1 EN UBUNTU 9.04

Esta guía pretende seguir paso a paso la instalación de RTAI 3.7.1 y RTnet 0.9.11 en un equipo con Ubuntu 9.04 como sistema operativo. Es preciso aclarar que la instalación de RTAI depende de la compilación de una versión limpia del kernel de Linux y por lo tanto dependerá también de las características del equipo utilizado, como consecuencia de lo anterior serán descritos los pasos necesarios para una adecuada instalación de RTAI además de los específicos según el equipo utilizado.

El equipo utilizado para la instalación tiene las siguientes características:

- Procesador Intel Core 2 Duo de 2Ghz
- RAM: 3GB
- Sistema Operativo: Ubuntu 9.04 (32bits)

Antes de iniciar, es necesario tener instalados los siguientes paquetes:

En general: subversión, build-essential, bootcd-mkinitramfs, make, gcc4.33, g++4.33 (se manejará la versión 4.33 del compilador por su compatibilidad con RTAI 3.7. En caso de querer instalar una versión de RTAI menor debe utilizarse una versión menor del compilador).

Para la compilación del kernel: kernel-package, linux-source, libncurses5-dev.

Para la instalación de rtai: libtool, automake.

Debe tenerse en cuenta que casi todos los comandos requieren un ingreso como root, por lo tanto, antes de todo, ingresar:

```
sudo su
```

1. DESCARGAS

a. Descargar el kenel de linux 2.6.28.9 (Vanilla Kernel) desde http://www.kernel.org/pub/linux/kernel/v2.6/
b. Descargar RTAI 3.7.1 desde https://www.rtai.org/index.php?&MMN_position=1:1
c. Descargar RTnet 0.9.11 (rtnet-0.9.11.tar.bz2) desde http://www.rtnet.org/download.html
d. Ubicar el kernel y rtnet en /usr/src/, y el paquete de RTAI en /opt/

```
cp <RUTA-DESCARGA-KERNEL-LINUX>/linux-2.6.28.9.tar.gz  /usr/src/
cp <RUTA-DESCARGA-RTAI>/rtai-3.7.1.tar.bz2  /opt/
cp <RUTA-DESCARGA-RTNET>/rtnet-0.9.11.tar.bz2  /usr/src
```

e. Desempaquetar

```
cd /usr/src/
tar xfv linux-2.6.28.9.tar.gz
```

```
tar xvjf rtnet-0.9.11.tar.bz2
cd /opt/
tar xfv rtai-3.7.1.tar.bz2
```

En la carpeta /opt/rtai-3.7.1/base/arch/x86/patches puede encontrarse los parches para el kernel. En esta versión de RTAI se disponen de parches para el kernel 2.6.24, 2.6.25, 2.6.28.9 y 2.6.29.4, por lo tanto, teniendo en cuenta que el equipo en donde se llevará a cabo la instalación tiene como sistema operativo Ubuntu 9.04 y en consecuencia una versión del kernel 2.6.28.13, será utilizada la versión más aproximada a la ya instalada y es por esta razón que fue descargada la versión limpia 2.6.28.9 del kernel de Linux.

2. PARCHE Y CONFIGURACIÓN DEL KERNEL

a. Parchar el kernel:

```
patch -p1 < /opt/rtai/base/arch/x86/patches/hal-linux-2.6.28.9-x86-2.2-07.patch
```

b. Con el fin de que la configuración del kernel sea lo más similar posible a la ya instalada en el equipo, es recomendable realizar los siguientes pasos:

```
cp /boot/config-2.6.28-11-generic  ./.config
make oldconfig
# Enter a todo
```

c. Configuración

```
make xconfig
```

El anterior comando lanza un menú con varios parámetros modificables, los cuales fueron establecidos o cambiados de la siguiente manera:

Cambios necesarios para RTAI:

- Enable loadable module support > Module versioning support = no
- Processor type and features > Interrupt pipeline = sí
- Processor type and features > Subarchitecture Type = PC-compatible

Debido a que el gestor de potencia es un obstáculo para obtener una buena latencia, deshabilitar:

- Power management options > Power Management support = no
- Power management options > CPU Frequency scaling > CPU Frequency scaling = no

Para que el nuevo kernel tenga un nombre entendible y acorde con la instalación, es recomendable colocarle un sufijo:

General setup > Local version - append to kernel release = -rtai-3.7.1

Para especificar el tipo de procesador (si no es conocido con seguridad que arquitectura maneja el procesador, es recomendable colocar 386):

Processor type and features > Processor family = 386

Con un procesador de núcleo simple deshabilite la siguiente opción. Si por el contrario se tiene un procesador que maneje varios núcleos la siguiente opción puede ser habilitada o deshabilitada según preferencia:

Processor type and features > Multi core scheduling = sí

Los siguientes parámetros se han establecido de la siguiente manera por necesidad particular del equipo:

- Processor type and features > symmetric multiprocessing = no
- Networking support > Bluetooth subsystem support = no
- Networking Support > IPv6 = no
- Kernel Hacking > Compile the kernel with debug info = no
- Device Drivers > sound card support > ISA, SPI, PCMCIA = no

3. COMPILACIÓN DEL KERNEL

```
make clean
make
make modules_install
make install
```

El comando make modules_install crea un módulo en la carpeta /lib/modules/ con el nombre <VERSIÓN-KERNEL-PARCHADO>-rtai-3.7.1, para este caso particular el nombre es 2.6.28.9-rtai-3.7.1

4. INSTALACIÓN DEL KERNEL

a. Crear la imagen del kernel parchado para RTAI.

```
cd /boot
mkinitramfs -o “initrd.img-NOMBRE DE LA NUEVA IMAGEN” “modulo creado en /lib/modules”
mkinitramfs -o initrd.img-2.6.28.9-rtai-3.7.1 2.6.28.9-rtai-3.7.1
```

2.6.28.9-rtai-3.7.1 es el modulo creado en el paso anterior.

b. Para que al reiniciar el sistema aparezca la opción del nuevo kernel, es necesario actualizar el archivo menu.lst ubicado en /boot/grub/ con el siguiente comando:

```
sudo update-grub
```

c. Reiniciar el sistema e ingresar por el kernel con el parche aplicado.

5. INSTALACIÓN DE RTAI

```
cd /usr/src/
ln -snf linux-2.6.28.9 linux
```

a. Crear una carpeta build dentro de la carpeta de RTAI, configurar y compilar:

```
cd /opt/rtai-3.7.1
mkdir build
cd build
make -f ../makefile xconfig
```

Configurar de la siguiente manera:

- General > Linux source tree =/usr/src/linux
- Machine (x86) > Number of CPUs (SMP-only) = 1
- Base System > Supported Services > Net RPC > Use Rtnet = sí
- Add-ons > Real-Time Driver Model over RTAI = yes

En la última opción RTAI ofrece un *framework* de C++ para desarrollar aplicaciones de tiempo real con este lenguaje de programación. El módulo encargado será denominado rtai_cpp.o.

Una vez estén configurados satisfactoriamente los anteriores parámetros, salir de la ventana de configuración aceptando las condiciones para guardar. Posteriormente, RTAI será compilado automáticamente.

b. Si no existen errores en la compilación, instalar RTAI:

```
make install
```

c. Realizar un respaldo de los archivos de dispositivos de RTAI:

```
cp -a /dev/rtai_shm /lib/udev/devices/
cp -a /dev/rtf[0-9] /lib/udev/devices/
```

d. Establecer la disponibilidad de las librerías de RTAI en el sistema.

```
echo /usr/realtime/lib/ > /etc/ld.so.conf.d/rtai.conf
exit
sudo ldconfig
```

e. Agregar a la variable $PATH del sistema el directorio /bin/ de RTAI agregando el final del archivo .profile ubicado en ~/ (como usuario) y/o /root/ (como root), lo siguiente:

```
PATH="$PATH:/usr/realtime/bin"
```

6. TEST DE RTAI

Para asegurarse de que la instalación de RTAI ha sido correcta, ejecutar el test de latencia de la siguiente manera:

```
cd /usr/realtime/testsuite/kern/latency/
sudo ./run
```

La consola debe mostrar algo similar a:

```
*
*
* Type ^C to stop this application.
*
*

## RTAI latency calibration tool ##
# period = 100000 (ns)
# avrgtime = 1 (s)
# do not use the FPU
# start the timer
# timer_mode is oneshot

RTAI Testsuite - KERNEL latency (all data in nanoseconds)
RTH|   lat min|   ovl min|   lat avg|   lat max|   ovl max|   overruns
RTD|       0|       0|    17494|    238019|    238019|       2
RTD|    7543|       0|    17414|    26819|    238019|       2
RTD|    8381|       0|    17457|    26819|    238019|       2
RTD|    8381|       0|    17421|    26819|    238019|       2
RTD|    8381|       0|    17519|    26819|    238019|       2
RTD|    8381|       0|    17545|    34362|    238019|       2
RTD|    8381|       0|    17579|    32686|    238019|       2
RTD|    8381|       0|    17640|    35200|    238019|       2
RTD|    8381|       0|    17635|    30171|    238019|       2
RTD|    8381|       0|    17502|    29333|    238019|       2
RTD|    8381|       0|    17492|    28495|    238019|       2
RTD|    8381|       0|    17390|    26819|    238019|       2
RTD|    8381|       0|    17450|    26819|    238019|       2
RTD|    8381|       0|    17403|    26819|    238019|       2
RTD|    8381|       0|    17436|    26819|    238019|       2
RTD|    8381|       0|    17393|    26819|    238019|       2
RTD|    8381|       0|    17446|    26819|    238019|       2
RTD|    8381|       0|    17409|    26819|    238019|       2
```

ANEXO B

ANEXO B. INTERFAZ PARA APLICACIONES DE TIEMPO REAL

Esta herramienta para desarrollo de aplicaciones con restricciones de tiempo real más conocida como RTAI[1], por sus siglas en ingles, no es propiamente un sistema operativo de tiempo real (RTOS[2]), por el contrario es una extensión del *kernel* de Linux que añade las capacidades de los sistemas operativos de tiempo real, con unos mínimos cambios sobre el sistema original. Fue desarrollado por el *Dipartimento di Ingeneria Aerospaziale, Politecnico di Milano*, por el profesor Paolo Mantegazza como una variante al RTLinux original, dado que este último no tenia soporte para punto flotante ni planificación periódica.

RTAI introduce una pequeña "capa" bajo el núcleo original de Linux, sirviendo de interfaz entre este último y el hardware del sistema, por lo que intercepta las interrupciones y llamadas que el *kernel* envía, controlando así, el instante en que son atendidas, dando prioridad a las tareas de tiempo real que se ejecutan junto con el *kernel* de Linux.

El planificador trata al *kernel* del sistema operativo como una tarea "ociosa" que solo es ejecutada si no hay tareas de tiempo real para ejecutarse y el núcleo de tiempo real está inactivo. La tarea de Linux nunca puede bloquear interrupciones o prevenir que sea expulsada por una tarea de mayor prioridad[3] (en este caso, cualquiera que sea marcada como de tiempo real), lo que previene se agregue latencia a las tareas de tiempo real. [1]

Así como en RTLinux, RTAI hace pocas variaciones en el kernel, gracias a que realiza una emulación software del hardware de control de interrupciones. Cuando ocurre una interrupción, el núcleo de tiempo real la intercepta y decide que despachar, si existe un manejador de tiempo real para dicha interrupción, este es invocado. Si, por el contrario no existe o dicho manejador expresa que desea compartir la interrupción con Linux, entonces esta es marcada como pendiente. Si Linux solicita que una interrupción sea habilitada, cualquier interrupción pendiente es habilitada y el manejador de Linux apropiado es invocado, con las interrupciones físicas re-activadas.

La extensión de Linux de tiempo real desacopla los componentes del núcleo de tiempo real de aquellos del núcleo de Linux para propósitos generales para que cada uno pueda ser optimizado de manera independiente y el núcleo de tiempo real se mantenga pequeño y simple.

Fundamentalmente, RTAI, RLINUX y las aplicaciones escritas para tomar las ventajas de estos operan del mismo modo. El núcleo de tiempo real, sus componentes y las aplicaciones RT corren todas en el espacio del *kernel* como módulos. Cada módulo cargado se inicializa listo para la operación del sistema. Estos módulos del *kernel* pueden ser cargados y removidos dinámicamente, ya sea por una aplicación o haciendo uso de las operaciones de carga de Linux.

---

[1] Real Time Application Interface

[2] Real Time Operative System

[3] Este mecanismo es conocido como *preemption* (6)

Las ventajas de ejecutar el sistema de tiempo real en el espacio del kernel de Linux es que minimiza el tiempo de conmutación entre tareas. Otra ventaja es la modularidad que ofrece la característica de desarrollar por módulos. Por ejemplo, si el planificador no es el adecuado para una aplicación en particular, entonces, dicho módulo puede ser reemplazado por uno que cumpla con las necesidades de la misma.

Una de las principales desventajas de ejecutar una tarea en el espacio del kernel de Linux es que un defecto de programación en una tarea de tiempo real puede colapsar el sistema entero, dado que no existe espacio de memoria protegida separado para una tarea individual.

Algunas de las capacidades que ofrece RTAI incluyen [1]:

- Compatibilidad con la extensión POSIX 1003.1c (Pthreads, incluyendo mutex y variables condicionales).
- Compatibilidad con la extensión POSIX 1003.1b (solo Pqueues).
- Mecanismos tradicionales de comunicación entre procesos[4] que incluye: semáforos, buzones (mailbox), FIFOs, memoria compartida y RPC[5].
- Asignación de memoria dinámica no bloqueante en el dominio de tiempo real
- Soporte para lenguaje PERL para planificar tareas de tiempo real suave.
- Interfaz /proc que ofrece información sobre las tareas, módulos, servicios y procesos de tiempo real, extendiendo el soporte estándar /proc de Linux.
- Módulo LXRT que permite el uso de llamadas de “RTAI” al sistema desde el espacio de usuario
- Soporte para arquitecturas de un solo procesador y multi-procesador
- Soporte para punto flotante
- Planificadores periódicos o de un solo disparo

Al igual que Linux, RTAI es *open source* por lo que puede ser descargado de manera gratuita y el núcleo de Linux puede ser parchado manualmente.

RTAI ofrece una alternativa gratuita y eficaz frente a los sistemas operativos de tiempo real más conocidos, debido a su facilidad de instalación, el bajo nivel de modificación al código original del núcleo de Linux, las potentes capacidades de la herramienta, el costo nulo, lo que facilita el uso en ambientes educativos y el soporte continuo, de la comunidad y los desarrolladores por lo que se recomienda ampliamente el uso del mismo.

[4] IPC, por sus siglas en ingles
[5] Llamada a procedimientos remotos

ANEXO C

ANEXO C. MOBILINUX

Desarrollado por Montavista, Mobilinux constituye tanto un sistema operativo Linux optimizado como una poderosa plataforma de desarrollo para dispositivos móviles tales como teléfonos celulares, dispositivos GPS, dispositivos médicos portables, terminales inalámbricos en puntos de venta, etc [2]. Este software está presente en más de 35 millones de dispositivos, muy lejos de cualquier otra alternativa comercial basada en Linux.

La compañía Montavista, de acuerdo a sus representantes, es el proveedor líder de Linux para infraestructuras de telecomunicaciones y dispositivos inteligentes. Distribuye sistemas operativos Linux de grado comercial, herramientas de desarrollo, soporte de expertos, diseño y migración de servicios. Su fundador, James Ready, tiene más de 25 años de experiencia en el campo y fue uno de los creadores del primer sistema operativo de tiempo real, VRTX. [3]

Algunas de las características principales del Mobilinux en cuanto al rendimiento son [4]:

- Duración de carga de la batería: Posee un sistema de gestión dinámico de potencia, totalmente configurable que incluye APIs que extienden la vida de la batería especialmente para aplicaciones con alto consumo de potencia, como por ejemplo multimedia.[6]
- Conectividad integrada: Posee soporte integrado para diversos dispositivos incluyendo SDIO, WiLAN/WiFi sobre USB, bluetooth sobre USB además de drivers para soporte de audio ALSA, entre otros.
- Rápido arranque: Típicamente los teléfonos con Mobilinux inician en menos de 5 segundos y están listos para realizar llamadas en menos de 10 segundos.
- Baja ocupación de memoria: La versión 5.0[7] puede ser implementada en menos de 2MB[8] para algunos dispositivos con capacidades restringidas y en menos de 14MB en teléfonos con las funcionalidades básicas.
- Seguridad robusta: El mecanismo de seguridad denominado uSELinux, una versión compacta del sistema SELinux[9] [5] desarrollado por la NSA[10] que asegura la confidencialidad de los mensajes y la integridad de los archivos y del software del sistema.
- Soporte para procesadores multinúcleo: Mobilinux 5.0 soporte dispositivos construidos con múltiples chips de procesamiento, por ejemplo, aquellos teléfonos que utilizan un procesador para banda base y otro para las aplicaciones del sistema, así como chips integrados de multiprocesamiento síncrono y asíncrono.

---

[6] Para más información, consultar el texto "Dynamic Power Manager Programmers'Guide" disponible en la web [7]

[7] Última versión de lanzada al mercado y con soporte hasta el 2010

[8] Megabytes

[9] Security-Enhanced Linux

[10] Agencia de Seguridad Nacional de los Estados Unidos, por sus siglas en ingles

Es importante resaltar que existe una mejora importante en el sistema de desarrollo, en palabras de su fundador “Ningún otro sistema operativo móvil existente en el mercado permite desatar la originalidad de los desarrolladores añadiendo nuevas funcionalidades de tantas formas. Este es el mejor SO móvil del mundo”. Todo gracias al primer sistema de desarrollo que incorpora KGDB (Kernel GNU Debugging) por USB. Nuevas aplicaciones para el desarrollo y depuración del kernel y aplicaciones para el SO. Su kit de desarrollo consta de un complejo IDE multisistema basado en Eclipse que provee al desarrollador de todas las herramientas para sacar sus productos al mercado lo más rápido posible [3]. La figura 1 muestra la plataforma de desarrollo, desde el nivel más bajo hasta el nivel de aplicación.

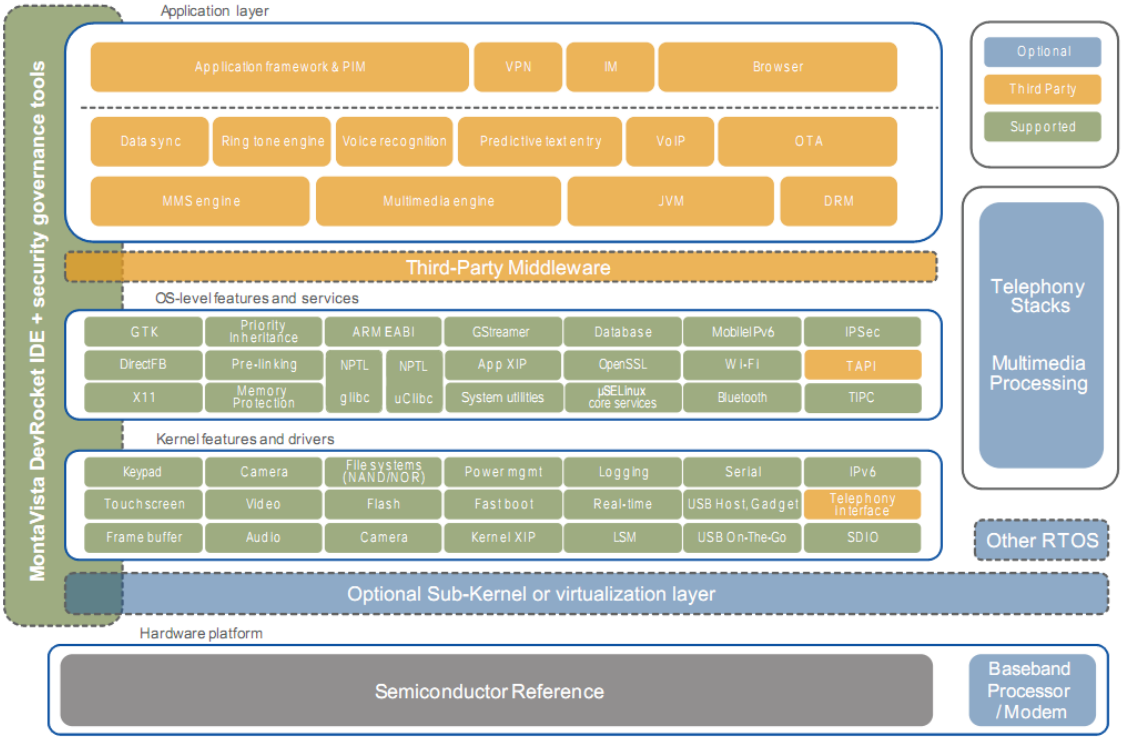

Figura 1. Plataforma de desarrollo de Mobilinux 5.0

Las capacidades y características de tiempo real que ofrece la plataforma son:

- Soporte para el mecanismo de *preemption* [6] en el kernel.
- Planificador de tiempo real.
- Temporizadores POSIX de alta resolución.
- Herencia de prioridad a nivel de aplicación.
- Encolado por prioridad.
- MutEx robustos.
- FutEx[11].

[11] MutEx en el espacio de usuario de rápida acción

Finalmente podemos ver que Mobilinux es un importante referente para los sistemas de desarrollo para aplicaciones móviles que la industria debe tener en cuenta ya que ofrece un alto desempeño desde el punto de vista del rendimiento y puede acortar los ciclos y costos de desarrollo dado el soporte incluido viéndose afectado de manera positiva y mostrando un retorno de la inversión hecha por las compañías desarrolladoras fácil y rápidamente.

BIBLIOGRAFÍA

[1] DIAPM ; Lineo Inc., *RTAI Programming Guide 1.0.* Lindon, 2000.

[2] MontaVista Software, Inc, “Commercial Linux Development for Wireless Handset and Mobile Devices from MontaVista ”, *montavista.* Disponible en web: http://www.mvista.com/product_detail_mob.php. [Citado: Noviembre 5, 2009]

[3] Araujo, M. “Mobilinux Un Linux para móviles y embebidos”, *Infolinuxblog.* Disponible en web: http://www.infolinuxblog.com/linux/mobilinux-un-linux-para-moviles-y-embebidos. [Citado: Noviembre 4, 2009]

[4] MontaVista Software, Inc, “Mobilinux datasheet”, *montavista.* Disponible en web: http://www.mvista.com/download/MontaVista-Mobilinux-5-datasheet.pdf. [Citado: Noviembre 5, 2009]

[5] National Security Agency. “Security-Enhanced Linux”, *NSA Research.* Disponible en web: http://www.nsa.gov/research/selinux/index.shtml. [Citado: Noviembre 5, 2009]

[6] Tics Realtime**.** “Tics Realtime Definitions”, *TICS REALTIME*. Disponible en web: http://www.concentric.net/~tics/ticsdef.htm#priorityBased**.** [Citado: Noviembre 6, 2009]

[7] MontaVista Software, Inc, “Embedded Linux Resource Download Library”, *montavista.* Disponible en web: http://www.mvista.com/download/topic.php?t=16. [Citado: Noviembre 5, 2009]