Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Engenharia Elétrica
ENG04008 Sistemas de Tempo Real

RTAI
Instalação

Prof. Walter Fetter Lages

1 de outubro de 2008

1 Pacotes Necessários

RTAI: Código-fonte do RTAI, disponível em `<https://www.rtai.org/RTAI>`. É aconselhável utilizar a ultima versão disponível, atualmente a versão 3.6.1.

***Kernel*:** Deve-se utilizar um *kernel* original[1], disponível em `<ftp://ftp.kernel.org/pub/linux/kernel/v2.6>`. O melhor é utilizar o *kernel* mais recente suportado pelo RTAI. Veja na seção 3 como determinar as versões do *kernel* suportadas pela versão do RTAI em questão.

2 Organização dos Pacotes

Usualmente o código-fonte do *kernel* fica em `/usr/src/linux-<versao>`, onde `<versao>` corresponde ao número da versão do *kernel*. Assim, pode-se ter no computador o código-fonte de diversas versões do *kernel*. Também é usual existir em `/usr/src` um *link* `linux` apontando para o diretório onde está a versão do *kernel* em uso.

Da mesma forma, é usual ter-se o código-fonte do RTAI em `/usr/src/rtai-<versao>` e ter-se um link `rtai` apontando para o diretório onde está a versão do RTAI em uso.

[1] Algumas distribuições vem com *kernel* modificado. É aconselhável baixar um original.

3 *Patch* do *Kernel*

Para funcionar, o RTAI utiliza um *pipeline* de interrupções, que é instalado através de um *patch* no *kernel* do Linux. Os *patches* para as versões suportadas do *kernel* estão disponíveis em `rtai/base/arch/<arquitetura>/patches`.

As versões do *kernel* suportadas pelo RTAI pode ser determinadas a partir do arquivos de *patch* em `rtai/base/arch/<arquitetura>/patches`, onde `<arquitetura>` é uma das arquiteturas suportadas pelo RTAI:

arm: processador ARM

i386: processadores Intel, utilizando Linux com arquitetura de 32 bits

ppc: processadores Power-PC

x86_64: processadores Intel, utilizando Linux com arquitetura de 64 bits

x86: processadores Intel, utilizando Linux com arquitetura unificada de 32/64 bits[2]

Após realizar o *patch*, é comum alterar-se o nome do diretório onde o Linux está instalado, para que o mesmo não seja confundido com um *kernel* original.

4 Configuração *kernel*

Esta é a etapa mais *tricky* do processo. Configurar o *kernel* significa escolher quais sub-sistemas, *drivers*, sistemas de arquivos, etc, serão compilados e farão parte do arquivo de *boot* ou serão compilados como módulos ou que não serão compilados porque nunca serão utilizados. Para fazer esta configuração de forma adequada é necessário conhecer o *hardware* da máquina onde o *kernel* vai executar e quais serviços do *kernel* o *software* que será executado necessitará. Obviamente, isto requer um com conhecimento do funcionamento do Linux e dos dispositivos de *hardware* instalados.

Uma alternativa é copiar a configuração do *kernel* que está executando na máquina e fazer as alterações necessárias. Esta configuração está disponível no pseudo-arquivo `/proc/config` ou na forma compactada no arquivo `/proc/config.gz`. No entanto, nem todas as distribuições habilitam esta funcionalidade no *kernel* que instalam. Eventualmente, a distribuição pode ter compilado esta opção como um módulo. Neste caso, para ter-se acesso à configuração é necessário carregar o módulo `configs.ko`.

[2] Esta arquitetura só existe a partir da versão 2.6.24 do *kernel*.

Adicionalmente, as distribuições configuram o *kernel* com inúmeras funcionalidades que dificilmente serão utilizadas na maioria dos sistemas. Embora esta funcionalidade sejam em sua maioria compiladas como módulos e portanto apenas ocupem espaço em disco, a compilação de todos estes módulos faz com que a compilação do *kernel* torne-se bastante demorada. Assim, o ideal é configurar o *kernel* para que sejam compilados apenas os módulos que serão utilizados, ou que tem alguma possibilidade de virem a ser utilizados.

A forma mais robusta de alterar a configuração do *kernel* é através do comando `make menuconfig`. Este comando executa um aplicativo com interface baseada no `ncurses` que permite a configuração do *kernel* através de *menus* hierárquicos. Por *default*, a configuração do *kernel* é salva no arquivo `.config` dentro da árvore do *kernel*, que é utilizado pelas etapas subseqüentes de compilação.

Uma boa idéia é manter-se um arquivo com a configuração do *kernel* em um diretório fora da árvore do *kernel*, de forma que ele não seja destruído acidentalmente ao trocar-se o código-fonte do *kernel* por outra versão.

Em especial para configurar um *kernel* para uso com o RTAI, deve-se habilitar a opção "Interrupt pipeline"no sub-menu "Processor type and features".

Outra detalhe importante na configuração do *kernel* é o método que será utilizado para carregar os *drivers* dos dispositivos necessários durante o *boot*, como, por exemplo os *drivers* do controlador de HD e do sistema de arquivo utilizado. Existem, basicamente duas formas de carregar estes drivers: Configurar o *kernel* para linkar estes *drivers* no arquivo de *boot* ou configurar estes *drivers* como módulos e inclui-los no `initrd` (disco de RAM), de onde serão carregados durante o *boot*.

Normalmente as distribuições utilizam o `initrd`, pois o *kernel* deve ser capaz de dar *boot* em diversas máquinas com configurações desconhecidas. Para facilitar a criação do `initrd` as distribuições incluem *scripts*, usualmente com o nome `mkinitrd`. Verifique nos *man pages* da sua distribuição a sintaxe correta para utilizar o *script* `mkinitrd`.

Aqui, será utilizado o método de linkar os *drivers* diretamente no arquivo de *boot*, por ser mais simples.

5 Compilação do *kernel* e dos Módulos

Esta etapa é relativamente simples, embora demorada. Basta executar os comandos para compilar o *kernel* e os módulos e esperar. Após a compilação o arquivo de *boot* do *kernel* estará em `arch/<arquitetura>/boot/bzImage`. Note que os módulos podem ser compilados independentemente do arquivo de *boot* do *kernel*. Assim, se for necessário reconfigurar o *kernel* para incluir outros módulos, basta recompilar os módulos.

6 Instalação do *kernel* e Módulos

Os arquivos de *boot* do *kernel* ficam em `/boot`. Pode-se ter vários arquivos de *boot* neste diretório, de forma a poder escolher entre eles através de um gerenciador de *boot* como GRUB ou o LILO. Além do arquivo de *boot* é usual copiar-se para `/boot` o arquivo `System.map` (que contém a lista de símbolos do *kernel*) e o arquivo de configuração utilizado para compilar o *kernel*. Para diferenciar as diferentes versões destes arquivos inclui-se no nome do arquivo a versão do *kernel* correspondente, de forma que os nomes dos arquivos em `/boot` assumem a forma `bzImage-<versao>`, `System.map-<versao>` e `config-<versao>`.

Os módulos do *kernel* são instalados em /lib/<versao>.

7 Gerenciador de *Boot*

Os gerenciadores de *boot* mais comuns utilizados com o Linux são o LILO e o GRUB. O LILO é mais fácil de configurar do que o GRUB, mas em caso de erro e principalmente se for esquecido de executar o programa `lilo` após qualquer alteração de configuração, pode resultar em um sistema que não é mais capaz de dar *boot* a partir do HD. Neste caso, é necessário dar um *boot* por CD, DVD ou USB, montar as partições manualmente e reconfigurar corretamente o LILO.

Já o GRUB é um pouco mais difícil de configurar, mas depois de instalado, para qualquer alteração é só editar o arquivo `/boot/grub/menu.lst` e a nova configuração já estará operacional. Não e' preciso executar qualquer programa para instalar a alteração, como no LILO. Além disso, o GRUB permite que se edite a configuração na tela de escolha do *kernel* que vai dar *boot*. Assim, se houver algum erro pode-se editar a configuração durante o processo de *boot* e corrigir o problema. É mais prático e mais seguro na hora de modificar as opções de *boot*.

8 Configuração do RTAI

O processo de configuração do RTAI é semelhante ao do *kernel*. Existe um *script* para fazer esta configuração através de um sistema de *menus* hierárquicos baseado na biblioteca `ncurses`. Em especial, deve-se tomar o cuidado de habilitar o "One-shot timer mode", que vem desabilitado por *default* e não habilitar Use RTNet", "LTT support"e "LEDS-based debugging support", que necessitam de outros pacotes para serem utilizados. Também não se deve habilitar "In-kernel C++ support", que não é suportado nas versões do *kernel* a partir de 2.4.x. As demais opções não são muito críticas e podem ser deixadas no *default*.

9 Compilação do RTAI

De forma semelhante ao *kernel*, a compilação do RTAI é relativamente simples, basta executar o comando `make`. É normal surgirem alguns *warnings*.

10 Instalação do RTAI

Por *default* o RTAI é instalado em `/usr/realtime`. Antes de efetuar a instalação convém remover este diretório, para evitar que sejam misturados arquivos de uma versao*Mesmo a mistura de arquivos da mesma versão mas compilados em momentos diferentes, com opções diferentes de configuração podem gerar problemas.* do RTAI com outra. Para instalar o RTAI, basta executar `makeinstall`.

11 Teste do RTAI

A distribuição do RTAI inclui três programas de tese, cada um com duas versões, uma para ser executada no modo *kernel* e outra para ser utilizada no modo usuário (lxrt). Os programas de teste estão, respectivamente nos diretórios `/usr/realtime/testsuite/kern`e `/usr/realtime/testsuite/user`. Um dos programas testa a latência, outro a preempção e o terceiro, o chaveamento entre tarefas. Para utiliza-lo, basta entrar no diretório correspondente e executar o comando `run`. Nos teses de latência deve-se observar que o *overrun* deve ser igual a zero o tempo todo. Nos testes de preempção, os valores de *jitter* devem ficar dentro de limites aceitáveis. No teste de chaveamento é normal a saída no terminal parar após algumas linhas, no modo *kernel* é necessário pressionar ^{C} para abortar a execução, enquanto no modo usuário o programa termina normalmente. Nestes testes, o tempo de chaveamento entre tarefas é mostrado.

12 Instalação Passo-a-Passo

Nesta seção estão as instruções passo-a-passo para instalação do RTAI. É utilizada a versão 3.6.1 do RTAI, a versão 2.6.24 do *kernel* e supõe-se que a plataforma é um PC com Linux utilizando a arquitetura unificada de 32/64 bits (arquitetura *x86*). O nome da máquina é *lyapunov*. A partição `/dev/sda5` contém o `/` e a partição `/dev/sda3` contém o `/boot`.

Supõe-se que se está executando um *shell* de super-usuário:

1. Baixar o pacote do RTAI de <`https://www.rtai.org/RTAI/rtai-3.6.1.tar.bz2`> e colocar no diretório `/usr/src`:

```
cd /usr/src
wget --no-check-certificate https://www.rtai.org/RTAI/rtai-3.6.1.tar.bz2
```

2. Descompactar o pacote do RTAI, ajustar as permissões e criar o *link*:

```
bzcat rtai-3.6.1.tar.bz2 | tar -xv
chown -R root.root rtai-3.6.1
chmod -R go-w rtai-3.6.1
ln -s rtai-3.6.1 rtai
```

3. Verificar as versões do *kernel* suportadas:

```
ls rtai/base/arch/x86/patches
```

   O arquivo `hal-linux-2.6.24-x86-2.0-07.patch` revela que a versão do *kernel* mais recente suportada é a 2.6.24.

4. Baixar o pacote com o código-fonte do *kernel*:

```
wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.tar.bz2
```

5. Descompactar o código-fonte do *kernel* e ajustar as permissões:

```
bzcat linux-2.6.24.7.tar.bz2 | tar -xv
chown -R root.root linux-2.6.24
chmod -R go-w linux-2.6.24
```

6. Realizar o *patch* do *kernel*, alterar o nome do diretório e ajustar o *link*:

```
cd linux-2.6.24
cat ../rtai/base/arch/x86/patches/hal-linux-2.6.24-x86-2.0-07.patch | patch -p1
cd ..
mv linux-2.6.24 linux-2.6.24-ipipe
rm linux
ln -s linux-2.6.24-ipipe linux
```

7. Copiar a configuração do *kernel* que está executando:

```
zcat /proc/config.gz > lyapunov.conf
```

8. Configurar o *kernel*:

   (a) Executar o *script* de configuração:

```
cd linux
make menuconfig
```

(b) Carregar a configuração , selecionando a opção "Load an Alternate Configuration File"e o arquivo `../lyapunov.conf`

(c) No sub-menu "General setup":

   i. Alterar "Local version - append to kernel release"para -ipipe"

(d) No sub-menu "Enable loadable module support":

   i. Desabilitar "Module versioning support
   ii. Desabilitar "Source checksum for all modules"

   "

(e) No sub-menu "Processor type and features":

   i. Desabilitar "Tickless System (Dynamic Ticks)"
   ii. Desabilitar "High Resolution timer Suport"
   iii. Ajustar "Subarchitecture Type"para "PC-compatible"
   iv. Ajustar "Processor family"conforme o processador da máquina
   v. Desabilitar "Generic x86 support"
   vi. Desabilitar "HPET Timer Support"
   vii. Desabilitar "SMT (Hyperthreading) scheduler support"
   viii. Ajustar "Preemption Model"para "Preemptible Kernel (Low-Latency Desktop)"
   ix. Habilitar "Preempt The Big kernel Lock"
   x. Habilitar "Interrupt pipeline"
   xi. Ajustar "Timer frequency"para 1000Hz

(f) No sub-menu "Device Drivers"selecionar o *driver* do controlador de HD da máquina e configura-lo para ser linkado com o arquivo de *boot* do kernel. Usualmente este *driver* estará no sub-menu "Serial ATA (prod) and Parallel ATA (experimental) drivers". Note que no caso de utilizar-se controladores SATA ou PATA (ao invés do tradicional IDE), deve-se também configurar o *driver* para HD SCSI, já que o *driver SATA* utiliza a mesma estrutura do *driver* SCSI.

(g) No sub-menu "File systems"selecionar os *drivers* do sistema de arquivos das partições `/` e `/boot`, utilizadas durante o *boot*. Normalmente estas partições utilizam os sistemas de arquivo Ext3 ou Reiser.

(h) Salvar a configuração no arquivo `../lyapunov.conf` utilizando a opção "Save an Alternate Configuration File"

(i) Salvar a configuração no arquivo `.config`[3]

[3]Algumas versões do *script* de configuração não salvam corretamente o arquivo `.config` na saída. Assim, é aconselhável salva-lo explicitamente.

(j) Sair do *script* de configuração

9. Compilar o *kernel*:

```
make bzImage
```

10. Compilar os módulos do *kernel*:

```
make modules
```

11. Instalar os arquivos de *boot*, símbolos e configuração do *kernel*

```
cp arch/x86/boot/bzImage /boot/bzImage-2.6.24-ipipe
cp System.map /boot/System.map-2.6.24-ipipe
cp .config /boot/config-2.6.24-ipipe
```

12. Instalar os módulos

```
make modules_install
```

13. Atualizar a configuração do gerenciador de *boot*:

    (a) Se for utilizado o GRUB: Editar o arquivo `/boot/grub/menu.lst` e incluir[4]:

    ```
    title RTAI
        root(hd0,2)
        kernel /bzImage-2.6.24.7-ipipe root=/dev/sda5
    ```

    (b) Se for utilizado o LILO:

        i. Editar o arquivo `/etc/lilo.conf` e incluir as seguintes linhas:

        ```
        image = /boot/bzImage-2.6.24.7-ipipe
            root = /dev/sda5
            label = RTAI
        ```

        ii. Executar o comando:

        ```
        lilo
        ```

14. Rebootar o sistema:

[4]Dependendo das demais configurações do gerenciador de *boot* (interface gráfica, partição de resume, etc) pode ser conveniente incluir outras opções na linha `kernel`. Uma boa idéia é verificar as opções utilizadas nos *kernels* que já estavam configurados.

```
reboot
```

15. Ainda como super-usuário, configurar RTAI:

    (a) Executar o *script* de configuração:

    ```
    cd /usr/src/rtai
    make menuconfig
    ```

    (b) No sub-menu "General":

        i. Habilitar "Enable extended configuration mode"
        ii. Habilitar "Enable debug symbols in modules"
        iii. Habilitar "Enable debug symbols in user-space programs"

    (c) No sub-menu "Machine(x86)": Ajustar o número de CPUs, se for o caso

    (d) No sub-menu "Base system", "Scheduling options":

        i. Habilitar "Enable full priority inheritance"
        ii. Habilitar "Keep Linux task priority aligned to RTAI"
        iii. Habilitar "One-shot timer mode"

    (e) No sub-menu "Base system", "Supported services":

        i. Configurar todas as opções como módulo, exceto "Use RTNet", que deve permanecer desabilitada
        ii. Habilitar "Support for Posix CLOCK_REALTIME APIs"

    (f) No sub-menu "Base system", "Other features":

        i. Habilitar "New return values of blocking RTAI APIs"
        ii. Habilitar "Mathfuns support in kernel"
        iii. Habilitar "C99 standard suport"
        iv. Desabilitar "Use vmalloc() support"

    (g) No sub-menu "Add-ons":

        i. Habilitar "Real-Time Driver Model over RTAI"
        ii. Habilitar "Shared interrupts"
        iii. Habilitar "Enable some elementary debugging messages"
        iv. No sub-menu "Drivers", habilitar "RTDM based 16550A driver support"

    (h)

    (i) Salvar a configuração no arquivo `../rtai.conf` utilizando a opção "Save an Alternate Configuration File"

(j) Salvar a configuração no arquivo `.config`

(k) Sair do *script* de configuração

16. Compilar o RTAI:

```
make
```

17. Instalar o RTAI:

```
rm -r /usr/realtime
make install
```

18. Testar o RTAI:

(a) Teste de latência no modo *kernel*. O *overrun* deve ser sempre zero.

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

(b) Teste de preempção no modo *kernel*. Os *jitters* devem ficar em valores razoáveis.

```
cd ../preempt
./run
```

(c) Teste de chaveamento no modo *kernel*. Os tempo se chaveamento de tarefas devem ficar em valores razoáveis.

```
cd ../switches
./run
```

(d) Teste de latência no modo usuário. O *overrun* deve ser sempre zero.

```
cd ../../user/latency
./run
```

(e) Teste de preempção no modo usuário. Os *jitters* devem ficar em valores razoáveis.

```
cd ../preempt
./run
```

(f) Teste de chaveamento no modo usuário. Os tempo se chaveamento de tarefas devem ficar em valores razoáveis.

```
cd ../switches
./run
```