Guia Foca GNU/Linux
Capítulo 16 - Kernel e Módulos

Este capítulo descreve em detalhes o que é o kernel, módulos, sua configuração e programas relacionados.

16.1 O Kernel

É a peça central do sistema operacional (o `Linux`), é ele que controla os dispositivos e demais periféricos do sistema (como memória, placas de som, vídeo, discos rígidos, disquetes, sistemas de arquivos, redes e outros recursos disponíveis). Muitos confundem isto e chamam a distribuição de sistema operacional. Isto é errado!
O *kernel* faz o controle dos periféricos do sistema e para isto ele deve ter o seu suporte incluído. Para fazer uma placa de som *Sound Blaster* funcionar, por exemplo, é necessário que o kernel ofereça suporte a este placa e você deve configurar seus parâmetros (como interrupção, I/O e DMA) com comandos específicos para ativar a placa e faze-la funcionar corretamente. Existe um documento que contém quais são os periféricos suportados/ não suportados pelo`GNU/Linux`, ele se chama `Hardware-HOWTO`.

Suas versões são identificadas por números como 2.2.30, 2.4.33, 2.6.23.6, as versões que contém um número par entre o primeiro e segundo ponto são versões estáveis e que contém números ímpares neste mesmo local são versões instáveis (em desenvolvimento). Usar versões instáveis não quer dizer que ocorrerá travamentos ou coisas do tipo, mas algumas partes do kernel podem não estar testadas o suficiente ou alguns controladores podem ainda estar incompletos para obter pleno funcionamento. Se opera sua máquina em um ambiente crítico, prefira pegar versões estáveis do kernel.

Após inicializar o sistema, o kernel e seus arquivos podem ser acessados ou modificados através do ponto de montagem `/proc`. Para detalhes veja O sistema de arquivos `/proc`, Seção 5.8.
Caso você tenha um dispositivo (como uma placa de som) que tem suporte no `GNU/Linux` mas não funciona veja Como adicionar suporte a Hardwares e outros dispositivos no kernel, Seção 16.3.

16.2 Módulos

São partes do kernel que são carregadas somente quando são solicitadas por algum aplicativo ou dispositivo e descarregadas da memória quando não são mais usadas. Este recurso é útil por 2 motivos: Evita a construção de um kernel grande (estático) que ocupe grande parte da memória com todos os drivers compilados e permite que partes do kernel ocupem a memória somente quando forem necessários.

Os módulos do kernel estão localizados no diretório `/lib/modules/versão_do_kernel/*` (onde `versão_do_kernel` é a versão atual do kernel em seu sistema, caso seja `2.6.23.6` o diretório que contém seus módulos será `/lib/modules/2.6.23.6`.

Os módulos são carregados automaticamente quando solicitados através do programa `kmod` ou manualmente através do arquivo `/etc/modules` , `insmod`ou `modprobe`. Atenção: Não compile o suporte ao seu sistema de arquivos raíz como módulo, isto o tornará inacessível, a não ser que esteja usando initrd.

16.3 Como adicionar suporte a Hardwares e outros dispositivos no kernel

Quando seu hardware não funciona mas você tem certeza que é suportado pelo `GNU/Linux`, é preciso seguir alguns passos para faze-lo funcionar corretamente:

- Verifique se o kernel atual foi compilado com suporte ao seu dispositivo. Também é possível que o suporte ao dispositivo esteja compilado como módulo. Dê o comando `dmesg` para ver as mensagens do kernel durante a inicialização e verifique se aparece alguma coisa referente ao dispositivo que deseja instalar (alguma mensagem de erro, etc). Caso não aparecer nada é possível que o driver esteja compilado como módulo, para verificar isto entre no diretório `/lib/modules/versao_do_kernel` e veja se encontra o módulo correspondente ao seu dispositivo (o módulo da placa *NE 2000* tem o nome de `ne.ko` e o da placa *Sound Blaster* de `sb.ko`, por exemplo).
  OBS: Nos kernel 2.4 e anteriores, a extensão dos módulos era `.o`.
  Caso o kernel não tiver o suporte ao seu dispositivo, você precisará recompilar seu kernel ativando seu suporte. Veja Recompilando o Kernel, Seção 16.11.
- Caso seu hardware esteja compilado no kernel, verifique se o módulo correspondente está carregado (com o comando `lsmod`). Caso não estiver, carregue-o com o `modprobe` (por exemplo, `modprobe sb io=0x220 irq=5 dma=1 dma16=5 mpuio=0x330`), para detalhes veja modprobe, Seção 16.8.
  O uso deste comando deverá ativar seu hardware imediatamente, neste caso configure o módulo para ser carregado automaticamente através do programa `modconf` ou edite os arquivos relacionados com os módulos (veja Arquivos relacionados com o Kernel e Módulos, Seção 16.12). Caso não tenha sucesso, será retornada uma mensagem de erro.

16.4 kmod

Este é o programa usado para carregar os módulos automaticamente quando são requeridos pelo sistema. Ele é um daemon que funciona constantemente fazendo a monitoração, quando verifica que algum dispositivo ou programa está solicitando o suporte a algum dispositivo, ele carrega o módulo correspondente.

Ele pode ser desativado através da recompilação do kernel, dando um `kill` no processo ou através do arquivo `/etc/modules` (veja /etc/modules, Seção 16.12.1. Caso seja desativado, é preciso carregar manualmente os módulos através do `modprobe` ou `insmod`.

16.5 lsmod

Lista quais módulos estão carregados atualmente pelo kernel. O nome `lsmod` é uma contração de `ls+módulos` - Listar Módulos. A listagem feita pelo`lsmod` é uma alternativa ao uso do comando `cat /proc/modules`.

A saída deste comando tem a seguinte forma:

```
Module Size Pages Used by

nls_iso8859_1 8000 1 1 (autoclean)

nls_cp437 3744 1 1 (autoclean)

ne 6156 2 1

8390 8390 2 [ne] 0
```

A coluna *Module* indica o nome do módulo que está carregado, a coluna *Used* mostra qual módulos está usando aquele recurso. O parâmetro *(autoclean)*no final da coluna indica que o módulo foi carregado manualmente (pelo `insmod` ou `modprobe`) ou através do `kmod` e será automaticamente removido da memória quando não for mais usado.
No exemplo acima os módulos *ne* e *8390* não tem o parâmetro *(autoclean)* porque foram carregados pelo arquivo `/etc/modules` (veja /etc/modules, Seção 16.12.1). Isto significa que não serão removidos da memória caso estiverem sem uso.
Qualquer módulo carregado pode ser removido manualmente através do comandos `rmmod`.

16.6 insmod

Carrega um módulo manualmente. Para carregar módulos que dependem de outros módulos para que funcionem, você duas opções: Carregar os módulos manualmente ou usar o `modprobe` que verifica e carrega as dependências correspondentes.
A sintaxe do comando é: `insmod [`*`módulo`*`] [`*`opções_módulo`*`]`

Onde:

módulo

É o nome do módulo que será carregado.

opções_módulo

Opções que serão usadas pelo módulo. Variam de módulo para módulo, alguns precisam de opções outros não, tente primeiro carregar sem opções, caso seja mostrada uma mensagem de erro verifique as opções usadas por ele. Para detalhes sobre que opções são suportadas por cada módulo, veja a sua documentação no código fonte do kernel em `/usr/src/linux/Documentation`

Exemplo: `insmod ne io=0x300 irq=10`

16.7 rmmod

Remove módulos carregados no kernel. Para ver os nomes dos módulos atualmente carregados no kernel digite `lsmod` e verifique na primeira coluna o nome do módulo. Caso um módulo tenha dependências e você tentar remover suas dependências, uma mensagem de erro será mostrada alertando que o módulo está em uso.
Exemplo: `rmmod ne`

16.8 modprobe

Carrega um módulo e suas dependências manualmente. Este comando permite carregar diversos módulos e dependências de uma só vez. O comportamento do `modprobe` é modificado pelo arquivo `/etc/modules.conf` .
A sintaxe deste comando é: `modprobe [módulo] [opções_módulo]`

Onde:

módulo

É o nome do módulo que será carregado.

opções_módulo

Opções que serão usadas pelo módulo. Variam de módulo para módulo, alguns precisam de opções outros não, tente primeiro carregar sem opções, caso seja mostrada uma mensagem de erro verifique as opções usadas por ele. Para detalhes sobre que opções são suportadas por cada módulo, veja a sua documentação no código fonte do kernel em `/usr/src/linux/Documentation`

Nem todos os módulos são carregados corretamente pelo `modprobe`, o `plip`, por exemplo, mostra uma mensagem sobre porta I/O inválida mas não caso seja carregado pelo `insmod`.
Exemplo: `modprobe ne io=0x300 irq=10`, `modprobe sb io=0x220 irq=5 dma=1 dma16=5 mpuio=0x330`

16.9 depmod

Verifica a dependência de módulos. As dependências dos módulos são verificadas pelos scripts em `/etc/init.d` usando o comando `depmod -a` e o resultado gravado no arquivo `/lib/modules/versao_do_kernel/modules.dep`. Esta checagem serve para que todas as dependências de módulos estejam corretamente disponíveis na inicialização do sistema. O comportamento do `depmod` pode ser modificado através do arquivo `/etc/modules.conf` . É possível criar a dependência de módulos imediatamente após a compilação do kernel digitando `depmod -a [versão_do_kernel]`.
Exemplo: `depmod -a`

16.10 modconf

Este programa permite um meio mais fácil de configurar a ativação de módulos e opções através de uma interface através de menus. Selecione a categoria de módulos através das setas acima e abaixo e pressione enter para selecionar os módulos existentes. Serão pedidas as opções do módulo (como DMA, IRQ, I/O) para que sua inicialização seja possível, estes parâmetros são específicos de cada módulo e devem ser vistos na documentação do código fonte do kernel no diretório `/usr/src/linux/Documentation`. Note que também existem módulos com auto-detecção mas isto deixa o sistema um pouco mais lento, porque ele fará uma varredura na faixa de endereços especificados pelo módulo para achar o dispositivo. As opções são desnecessárias em alguns tipos de módulos.
As modificações feitas por este programa são gravadas no diretório `/etc/modutils` em arquivos separados como `/etc/modutils/alias` - alias de módulos, `/etc/modutils/modconf` - opções usadas por módulos, `/etc/modutils/paths` - Caminho onde os módulos do sistema são encontrados. Dentro de `/etc/modutils` é ainda encontrado um sub-diretório chamado `arch` que contém opções específicas por arquiteturas.
A sincronização dos arquivos gerados pelo `modconf` com o `/etc/modules.conf` é feita através do utilitário `update-modules`. Ele é normalmente executado após modificações nos módulos feitas pelo `modconf`.

16.11 Recompilando o Kernel

Será que vou precisar recompilar o meu kernel? você deve estar se perguntando agora. Abaixo alguns motivos para esclarecer suas dúvidas:

- Melhora o desempenho do kernel. O kernel padrão que acompanha as distribuições `GNU/Linux` foi feito para funcionar em qualquer tipo de sistema e garantir seu funcionamento e inclui suporte a praticamente tudo. Isto pode gerar desde instabilidade até uma grade pausa do kernel na inicialização quando estiver procurando pelos dispositivos que simplesmente não existem em seu computador!

  A compilação permite escolher somente o suporte aos dispositivos existentes em seu computador e assim diminuir o tamanho do kernel, desocupar a memória RAM com dispositivos que nunca usará e assim você terá um desempenho bem melhor do que teria com um kernel pesado.

- Incluir suporte a alguns hardwares que estão desativados no kernel padrão (SMP, APM, ACPI, Virtualização, Firewall, Bridge, memory cards, drivers experimentais, etc).

- Se aventurar em compilar um kernel (sistema operacional) personalizado em seu sistema.

- Tornar seu sistema mais seguro

- Impressionar os seus amigos, tentando coisas novas.

Serão necessários uns 300Mb de espaço em disco disponível para copiar e descompactar o código fonte do kernel e alguns pacotes de desenvolvimento como o `gcc`, `cpp`, `binutils`, `gcc-i386-gnu`, `bin86`, `make`, `dpkg-dev`, `perl`, `kernel-package` (os três últimos somente para a distribuição `Debian`).

Na distribuição Debian, o melhor método é através do kernel-package que faz tudo para você (menos escolher o que terá o não o suporte no kernel) e gera um pacote .deb que poderá ser usado para instalar o kernel em seu sistema ou em qualquer outro que execute a Debian ou distribuições baseadas (Ubuntu, etc). Devido a sua facilidade, a compilação do kernel através do kernel-package é muito recomendado para utilizadores iniciantes e para aqueles que usam somente um kernel no sistema (é possível usar mais de dois ao mesmo tempo, veja o processo de compilação manual adiante neste capítulo). Siga este passos para recompilar seu kernel através do kernel-package:

1. Descompacte o código fonte do kernel (através do arquivo linux-2.6.XX.XX.tar.bz2) para o diretório /usr/src. Caso use os pacotes da Debian eles terão o nome de kernel-source-2.6.XX.XX, para detalhes de como instalar um pacote, veja Instalar pacotes, Seção 20.1.2.
2. Após isto, entre no diretório onde o código fonte do kernel foi instalado com cd /usr/src/linux (este será assumido o lugar onde o código fonte do kernel se encontra).
3. Como utilizador root, digite make config. Você também pode usar make menuconfig (configuração através de menus) ou make xconfig(configuração em modo gráfico) mas precisará de pacotes adicionais para que estes dois funcionem corretamente.
   Serão feitas perguntas sobre se deseja suporte a tal dispositivo, etc. Pressione Y para incluir o suporte diretamente no kernel, M para incluir o suporte como módulo ou N para não incluir o suporte. Note que nem todos os drivers podem ser compilados como módulos.
   Escolha as opções que se encaixam em seu sistema. se estiver em dúvida sobre a pergunta digite ? e tecle Enter para ter uma explicação sobre o que aquela opção faz. Se não souber do que se trata, recomendo incluir a opção (pressionando Y ou M. Este passo pode levar entre 5 minutos e 1 Hora (utilizadores que estão fazendo isto pela primeira vez tendem a levar mais tempo lendo e conhecendo os recursos que o GNU/Linux possui, antes de tomar qualquer decisão). Não se preocupe se esquecer de incluir o suporte a alguma coisa, você pode repetir o passo make config (todas as suas escolhas são gravadas no arquivo .config), recompilar o kernel e instalar em cima do antigo a qualquer hora que quiser.
4. Após o make config chegar ao final, digite make-kpkg clean para limpar construções anteriores do kernel.
5. Agora compile o kernel digitando make-kpkg --revision=teste.1.0 kernel-image. A palavra teste pode ser substituída por qualquer outra que você quiser e número da versão 1.0 serve apenas como controle de suas compilações (pode ser qualquer número).
   Observação: Não inclua hífens (-) no parâmetro --revision, use somente pontos.
6. Agora após compilar, o kernel será gravado no diretório superior (..) com um nome do tipo linux-image-2.6.23.6-i386_teste.1.0.deb. Basta você digitar dpkg -i kernel-image-2.6.23.6-i386_teste.1.0.deb e o dpkg fará o resto da instalação do kernel para você e perguntará se deseja criar um disquete de inicialização (recomendável).
7. Reinicie seu computador, seu novo kernel iniciará e você já perceberá a primeira diferença pela velocidade que o GNU/Linux é iniciado (você inclui somente suporte a dispositivos em seu sistema). O desempenho dos programas também melhorará pois cortou o suporte a dispositivos/funções que seu computador não precisa.

Caso alguma coisa sair errada, coloque o disquete que gravou no passo anterior e reinicie o computador para fazer as correções.

Para recompilar o kernel usando o método manual, siga os seguintes passos:

- Descompacte o código fonte do kernel (através do arquivo linux-2.6.XX.XX.tar.bz2) para o diretório `/usr/src`. O código fonte do kernel pode ser encontrado em ftp://ftp.kernel.org/.
- Após isto, entre no diretório onde o código fonte do kernel foi instalado com `cd /usr/src/linux` (este será assumido o lugar onde o código fonte do kernel se encontra).
- Como utilizador `root`, digite `make config`. Você também pode usar `make menuconfig` (configuração através de menus) ou `make xconfig` (configuração em modo gráfico) mas precisará de pacotes adicionais.

Serão feitas perguntas sobre se deseja suporte a tal dispositivo, etc. Pressione `Y` para incluir o suporte diretamente no kernel, `M` para incluir o suporte como módulo ou `N` para não incluir o suporte. Note que nem todos os drivers podem ser compilados como módulos.
Escolha as opções que se encaixam em seu sistema. se estiver em dúvida sobre a pergunta digite `?` e tecle Enter para ter uma explicação sobre o que aquela opção faz. Se não souber do que se trata, recomendo incluir a opção (pressionando `Y` ou `M`. Este passo pode levar entre 5 minutos e 1 Hora (utilizadores que estão fazendo isto pela primeira vez tendem a levar mais tempo lendo e conhecendo os recursos que o `GNU/Linux` possui antes de tomar qualquer decisão). Não se preocupe se esquecer de incluir o suporte a alguma coisa, você pode repetir o passo `make config`, recompilar o kernel e instalar em cima do antigo a qualquer hora que quiser.

- Caso esteja compilando um kernel 2.4 ou inferior, Digite o comando `make dep` para verificar as dependências dos módulos. Se estiver compilando um kernel 2.6 ou superior, pule esse comando.
- Digite o comando `make clean` para limpar construções anteriores do kernel.
- Digite o comando `make` para iniciar a compilação do kernel e seus módulos. Aguarde a compilação, o tempo pode variar dependendo da quantidade de recursos que adicionou ao kernel, a velocidade de seu computador e a quantidade de memória RAM disponível.

Caso tenha acrescentado muitos ítens no Kernel, é possível que o comando `make zImage` falhe no final (especialmente se o tamanho do kernel estático for maior que 505Kb). Neste caso use `make bzImage`. A diferença entre *zImage* e *bzImage* é que o primeiro possui um limite de tamanho porque é descompactado na memória básica (recomendado para alguns Notebooks), já a *bzImage*, é descompactada na memória estendida e não possui as limitações da *zImage*.

- A compilação neste ponto está completa, você agora tem duas opções para instalar o kernel: Substituir o kernel anterior pelo recém compilado ou usar os dois. A segunda questão é recomendável se você não tem certeza se o kernel funcionará corretamente e deseja iniciar pelo antigo no caso de alguma coisa dar errado.

Se você optar por substituir o kernel anterior:

- É recomendável renomear o diretório `/lib/modules/versão_do_kernel` para `/lib/modules/versão_do_kernel.old`, isto será útil para restauração completa dos módulos antigos caso alguma coisa der errado.
- Execute o comando `make modules_install` para instalar os módulos do kernel recém compilado em `/lib/modules/versão_do_kernel`.
- Copie o arquivo `zImage` que contém o kernel de `/usr/src/linux/arch/i386/boot/zImage` para `/boot/vmlinuz-2.XX.XX` (*2.XX.XX* é a versão do kernel anterior)
- Verifique se o link simbólico `/vmlinuz` aponta para a versão do kernel que compilou atualmente (com `ls -la /`). Caso contrário, apague o arquivo`/vmlinuz` do diretório raíz e crie um novo link com `ln -s /boot/vmlinuz-2.XX.Xx /vmlinuz` apontando para o kernel correto.
- Execute o comando `lilo` para gerar um novo setor de partida no disco rígido. Para detalhes veja LILO, Seção 6.1.
- Reinicie o sistema (`shutdown -r now`).
- Caso tudo esteja funcionando normalmente, apague o diretório antigo de módulos que salvou e o kernel antigo de `/boot`. Caso algo tenha dado errado e seu sistema não inicializa, inicie a partir de um disquete, apague o novo kernel, apague os novos módulos, renomeie o diretório de módulos antigos para o nome

original, ajuste o link simbólico `/vmlinuz` para apontar para o antigo kernel e execute o `lilo`. Após reiniciar seu computador voltará como estava antes.

Se você optar por manter o kernel anterior e selecionar qual será usado na partida do sistema (útil para um kernel em testes):

- Execute o comando `make modules_install` para instalar os módulos recém compilados do kernel em `/lib/modules/versao_do_kernel`.
- Copie o arquivo `zImage` que contém o kernel de `/usr/src/linux/arch/i386/boot/zImage` para `/boot/vmlinuz-2.XX.XX` (*2.XX.XX* é a versão do kernel anterior)
- Crie um link simbólico no diretório raíz (`/`) apontando para o novo kernel. Como exemplos será usado `/vmlinuz-novo`.
- Modifique o arquivo `/etc/lilo.conf` para incluir a nova imagem de kernel. Por exemplo:

```
Antes da modificação:

boot=/dev/hda

prompt

timeout=200

delay=200

map=/boot/map

install=menu

image = /vmlinuz

root = /dev/hda1

label = 1

read-only

Depois da modificação:

boot=/dev/hda

prompt

timeout=200

delay=200

map=/boot/map

install=menu

image = /vmlinuz

root = /dev/hda1

label = 1

read-only
```

```
image = /vmlinuz-new

root = /dev/hda1

label = 2

read-only
```

Se você digitar 1 no aviso de boot: do Lilo, o kernel antigo será carregado, caso digitar 2 o novo kernel será carregado. Para detalhes veja Criando o arquivo de configuração do LILO, Seção 6.1.1 e Um exemplo do arquivo de configuração lilo.conf, Seção 6.1.3.

- Execute o comando lilo para gravar o novo setor de boot para o disco rígido.
- Reinicie o computador
- Carregue o novo kernel escolhendo a opção 2 no aviso de boot: do Lilo. Caso tiver problemas, escolha a opção 1 para iniciar com o kernel antigo e verifique os passos de configuração (o arquivo lilo.conf foi modificado corretamente?.

Em alguns casos (como nos kernels empacotados em distribuições GNU/Linux) o código fonte do kernel é gravado em um diretório chamado kernel-source-xx.xx.xx. É recomendável fazer um link com um diretório GNU/Linux, pois é o padrão usado pelas atualização do código fonte através de patches (veja Aplicando Patches no kernel, Seção 16.13).

Para criar o link simbólico, entre em /usr/src e digite: ln -s kernel-source-xx.xx.xx linux.

Se quiser mais detalhes sobre a compilação do kernel, consulte o documento *kernel-howto*.

16.12 Arquivos relacionados com o Kernel e Módulos

Esta seção descreve os arquivos usados pelo kernel e módulos, a função de cada um no sistema, a sintaxe, etc.

16.12.1 /etc/modules

A função deste arquivo é carregar módulos especificados na inicialização do sistema e mantê-los carregado todo o tempo. É útil para módulos de placas de rede que precisam ser carregados antes da configuração de rede feita pela distribuição e não podem ser removidos quando a placa de rede estiver sem uso (isto retiraria seu computador da rede).

Seu conteúdo é uma lista de módulos (um por linha) que serão carregados na inicialização do sistema. Os módulos carregados pelo arquivo /etc/modulespode ser listados usando o comando lsmod (veja lsmod, Seção 16.5.

Se o parâmetro auto estiver especificado como um módulo, o kmod será ativado e carregará os módulos somente em demanda, caso seja especificadonoauto o programa kmod será desativado. O kmod é ativado por padrão nos níveis de execução 2 ao 5.

Ele pode ser editado em qualquer editor de textos comum ou modificado automaticamente através do utilitário modconf.

16.12.2 modules.conf

O arquivo `/etc/modules.conf` permite controlar as opções de todos os módulos do sistema. Ele é consultado pelos programas `modprobe` e `depmod`. As opções especificadas neste arquivo facilita o gerenciamento de módulos, evitando a digitação de opções através da linha de comando.
Note que é recomendado o uso do utilitário `modconf` para configurar quaisquer módulos em seu sistema e o utilitário `update-modules` para sincronização dos arquivos gerados pelo `modconf` em `/etc/modutils` com o `/etc/modules.conf` (geralmente isto é feito automaticamente após o uso do `modconf`). Por este motivo não é recomendável modifica-lo manualmente, a não ser que seja um utilizador experiente e saiba o que está fazendo. Veja modconf, Seção 16.10

Por exemplo: adicionando as linhas:

```
alias sound sb

options sb io=0x220 irq=5 dma=1 dma16=5 mpuio=0x330
```

permitirá que seja usado somente o comando `modprobe sb` para ativar a placa de som.

16.13 Aplicando Patches no kernel

Patches são modificações geradas pelo programa `diff` em que servem para atualizar um programa ou texto. Este recurso é muito útil para os desenvolvedores, pois podem gerar um arquivo contendo as diferenças entre um programa antigo e um novo (usando o comando `diff`) e enviar o arquivo contendo as diferenças para outras pessoas.
As pessoas interessadas em atualizar o programa antigo, podem simplesmente pegar o arquivo contendo as diferenças e atualizar o programa usando o`patch`.
Isto é muito usado no desenvolvimento do kernel do `GNU/Linux` em que novas versões são lançadas freqüentemente e o tamanho kernel completo compactado ocupa cerca de 18MB. Você pode atualizar seu kernel pegando um patch seguinte a versão que possui em ftp://ftp.kernel.org/.

Para aplicar um patch que atualizará seu kernel 2.6.23 para a versão 2.6.24 você deve proceder da seguinte forma:

- Descompacte o código fonte do kernel 2.6.23 em `/usr/src/linux` ou certifique-se que existe um link simbólico do código fonte do kernel para`/usr/src/linux`.
- Copie o arquivo `patch-2.6.24.gz` de ftp://ftp.kernel.org/ para `/usr/src`.
- Use o comando `gzip -dc patch-2.6.24|patch -p0 -N -E` para atualizar o código fonte em `/usr/src/linux` para a versão 2.6.24.
  Alternativamente você pode primeiro descompactar o arquivo `patch-2.6.24.gz` com o `gzip` e usar o comando `patch -p0 -N -E <patch-2.6.24`para atualizar o código fonte do kernel.
  O `GNU/Linux` permite que você obtenha o mesmo resultado através de diferentes métodos, a escolha é somente sua.

Caso deseja atualizar o kernel 2.6.20 para 2.6.24, como no exemplo acima, você deverá aplicar os patches em seqüência (do patch 2.6.20 ao 2.6.24). Vale a pena observar se o tamanho total dos patches ultrapassa ou chega perto o tamanho do kernel completo, pois dependendo da quantidade de alterações pode ser mais viável baixar diretamente a nova versão.