# Administrando Kernel no Debian GNU/Linux

Gustavo Noronha Silva <kov@debian.org>

0.7.1

## Resumo

O kernel é uma das partes centrais do Sistema Operacional. O Debian possui uma infra-estrutura extremamente poderosa para que os administradores de sistema preparem kernels e módulos para seus sistemas. A intenção desse manual é reunir em um lugar, de modo prático, a documentação dos processos envolvidos nessa estrutura.

## Nota de Copyright

# Sumário

**1 Introdução** **1**

1.1 Mas por que escrever esse manual? . . . . . 1

1.2 Do que preciso para começar? . . . . . 1

**2 Lidando com fontes de kernel e módulos** **3**

2.1 Como conseguir os fontes do kernel? . . . . . 3

2.2 Como conseguir fontes de módulos? . . . . . 3

2.3 Como conseguir patches? . . . . . 4

**3 Compilando o kernel** **5**

3.1 Como compilar o kernel? . . . . . 5

3.2 Como aplicar patches que estão empacotados? . . . . . 5

3.3 Como aplicar patches não empacotados? . . . . . 6

**4 Como compilar módulos de terceiros** **7**

4.1 Antes de compilar . . . . . 7

4.2 Compilando módulos de terceiros . . . . . 7

# Capítulo 1

# Introdução

## 1.1 Mas por que escrever esse manual?

Pelo mesmo motivo que escrevi os outros. O Debian sempre tem recursos e convenções muito poderosas. Muito mais que qualquer outro sistema que conheço, mas muitas vezes essas informações estão dispersas, difíceis de se encontrar.

É triste ver alguém com problemas para compilar um kernel, aplicar um patch, compilar módulos de terceiros porque estão usando o método convencional sabendo que o Debian facilita muito essas coisas.

## 1.2 Do que preciso para começar?

Você precisa do pacote `kernel-package` e todas as suas dependências. Procure instalar também os pacotes que o `kernel-package` recomenda e sugere, já que podem ser úteis em algumas situações. Além dele, para o final da compilação, o pacote `bin86` é necessário.

# Capítulo 2

# Lidando com fontes de kernel e módulos

## 2.1 Como conseguir os fontes do kernel?

No Debian há pacotes de várias versões do kernel, com nome `kernel-source-`*`versão`*. Você pode instalar um deles, o que vai colocar um arquivo `.tar.bz2` no `/usr/src`. Depois basta descompactar:

```
# cd /usr/src
# tar jxpvf kernel-source-versão.tar.bz2
```

Isso cria o diretório `/usr/src/kernel-source-`*`versão`*. Depois é bom criar um link simbólico para `/usr/src/linux`:

```
# cd /usr/src
# ln -s kernel-source-versão linux
```

Outra maneira de conseguir os fontes do kernel é baixando-os de um dos mirrors que distribuem os arquivos oficiais do kernel. Pode-se usar os procedimentos aqui documentados com eles, também.

## 2.2 Como conseguir fontes de módulos?

Há módulos de terceiros que podem ser úteis com o kernel. Dois exemplos são o ALSA e o `pcmcia-cs`. Há pacotes para eles no Debian, com nomes parecidos com *`nome`*`-source`. Basta instalar esses pacotes para ter arquivos `.tar.gz` em /usr/src. É necessário depois descomprimí-los. Usando o ALSA como exemplo:

```
# cd /usr/src
# tar zxpvf alsa-driver.tar.gz
```

Pode-se também usar o modo tradicional. Pegar os módulos das fontes oficiais para compilar normalmente depois da compilação dos módulos.

## 2.3 Como conseguir patches?

Há pacotes com nomes `kernel-patch-`*`nome`* que tornam fácil a inclusão dos patches em um kernel, mesmo que não se esteja usando o sistema Debian de compilação de kernel.

Os patches, depois de instalados, ficam em `/usr/src/kernel-patches` e não há necessidade de descomprimí-los.

# Capítulo 3

# Compilando o kernel

## 3.1 Como compilar o kernel?

Antes de mais nada veja 'Como conseguir os fontes do kernel?' on page 3.

Para compilar o kernel basta um simples comando, que faz todo o trabalho:

```
# make-kpkg --revision hostname.versão kernel-image
```

O `make-kpkg` é o nome do comando que lida com kernel no Debian. A opção `--revision` é muito importante. Sem ela as versões pré-compiladas do kernel do Debian irão provavelmente ser instaladas por cima da sua versão customizada. O jeito mais seguro de usá-la, embora não seja obrigatório, é colocar *hostname.versão*.

O parâmetro `kernel_image`, por fim, diz ao `make-kpkg` que deve construir o pacote do kernel e seus módulos. Ele é como um alvo de Makefile. Existem outros alvos, como `kernel_headers` que você pode usar para ter pacotes específicos que precisar. Olhe a página de manual do `make-kpkg` para mais informações.

## 3.2 Como aplicar patches que estão empacotados?

Os patches que você instalou podem ser aplicados durante a compilação. Ou você pode preferir aplicar você mesmo, ao invés de deixar esse trabalho para o `make-kpkg`.

Para fazer o `make-kpkg` aplicar, basta adicionar a opção `--added-patches` e definir a variável de ambiente `PATCH_THE_KERNEL` para `YES`. Dessa forma:

```
# PATCH_THE_KERNEL=YES make-kpkg --revision couve.1 \
   --added-patches debianlogo,mosix kernel_image
```

Note que você pode especificar diversos patches usando vírgulas para separá-los. Há uma maneira de fazer com que a variável de ambiente `PATCH_THE_KERNEL` não precise estar ligada. Basta colocar o seguinte no arquivo `/etc/kernel-pkg.conf`:

```
patch_the_kernel := YES
```

Para aplicar um patch "na mão" basta ir para o diretório do fonte do kernel e rodar o script `apply` daquele patch. Por exemplo, para aplicar o patch `debianlogo` ao kernel `2.4..18`:

```
# cd /usr/src/kernel-source-2.4.18
# /usr/src/kernel-patches/all/apply/debianlogo
```

Note que nem todos os patches estarão no subdiretório `all` do `kernel-patches`. Alguns patches que são específicos de arquitetura podem estar num diretório com o nome daquela arquitetura. No caso de computadores comuns, esse diretório pode ser `i386`.

Para saber como obter patches para o kernel veja 'Como conseguir patches?' on page 4.

## 3.3 Como aplicar patches não empacotados?

Basta aplicá-los normalmente, como se aplicaria qualquer outro patch em qualquer outro kernel. Vá ao diretório do kernel depois de descomprimir o arquivo `.tar.bz2` e aplique o patch.

# Capítulo 4

# Como compilar módulos de terceiros

## 4.1 Antes de compilar

Antes de compilar é sempre bom ver se os módulos exigem configuração especial. O ALSA não é um bom exemplo de módulo que exige configuração, mas é um exemplo de algo que é bom conhecer antes de começar.

Usando ele como exemplo, antes de mais nada instale o pacote `alsa-base` e, claro, o `alsa-source`. Dependendo de suas configurações do `debconf` você verá perguntas úteis para a compilação/configuração durante a instalação desses. Se não for esse o seu caso faça:

```
# dpkg-reconfigure alsa-base
(configuração)
# dpkg-reconfigure alsa-source
```

Procure ler o `README` dos módulos antes de compilá-los.

## 4.2 Compilando módulos de terceiros

Para compilar os módulos de terceiros, você pode simplesmente adicionar mais opções na linha de comando da compilação do kernel, ficando assim:

```
# make-kpkg --revision hostname.versão \
--added-modules alsa-drivers kernel_image modules_image
```

As mudanças são: a opção `--added-modules`, que pode ser seguida pelos nomes dos módulos que você quer compilar. Esses nomes devem ser nomes dos diretórios que estão dentro de `/usr/src/modules` e podem ser especificados separados por vírgulas. A opção `modules_image`, que compila e cria o pacote `.deb` dos módulos de terceiros.

Você pode fazer isso, também, como um processo independente, chamando apenas o comando `modules_image`, ao invés de compilar junto com o kernel. Note que é bom passar todas as opções que foram passadas para a compilação do kernel, já que isso pode afetar a compilação dos módulos. Principalmente, confira a `--revision`.