# Guia Prático para o Debian GNU/Linux

Gustavo Noronha Silva <kov@debian.org>

1.2

## Resumo

Esse é um guia prático para o Sistema Operacional Debian GNU/Linux. Ele pretende ser um manual de referência para funções úteis a às vezes pouco conhecidas.

## Nota de Copyright

# Sumário

# Capítulo 1

# A Documentação do Debian

## 1.1 O Sistema de Documentação (doc-base)

O Debian possui um sistema centralizado de documentação. O desenvolvedor do pacote deve "registrar" sua documentação nesse sistema e assim torná-la disponível para o sistema.

O `doccentral` é um dos programas que têm como função levar o usuário diretamente à documentação, usando o navegador especificado pelo usuário como padrão.

Antes de usá-lo é necessário instalar o pacote `doc-central`. Veja 'Como instalar pacotes' on page 3 para maiores informações. Instale também o pacote `doc-base`.

Para iniciá-lo, basta digitar em um prompt shell:

```
$ doccentral
```

O `doc-central` procura na variável de ambiente $BROWSER. Se não a encontra usa o navegador que encontrar. Por exemplo, para definir que o navegador a ser utilizado seja o `galeon`, coloca-se:

```
$ export BROWSER=galeon
```

no arquivo `/home/usuário/.bash_profile` e/ou no arquivo `/home/usuário/.bashrc`.

Assim, quando o `doc-central` for executado iniciará o galeon para navegar na documentação Debian.

## 1.2 Documentação do Debian via www

O `dwww` cria um ambiente www para **toda** a documentação disponível no sistema. Você pode até mesmo ler as manpages através de um browser, o que torna sua leitura extremamente mais agradável.

Para usá-lo instale o pacote `dwww` e, também o `doc-base` para ter melhores resultados, já que nem todos os pacotes se registram diretamente com o dwww.

Há duas maneiras de se usar o `dwww`: como um índice, no qual se navega até achar o que quer, ou como uma consulta rápida à toda a documentação de um pacote. Para ver o índice basta rodar:

```
$ dwww
```

Para ver toda a documentação disponível para um pacote, faça:

```
$ dwww pacote
```

Há outros tipos de programas para acessar a documentação indexada pelo `doc-base`, o `doc-central` por exemplo, que também usa um servidor http.

## 1.3 Documentação Online

O Debian possui ampla documentação na Internet. A principal fonte de documentação é o site do Debian: `http://www.debian.org`.

Há documentação disponível em várias línguas e existem vários projetos de localização do Debian. Portanto não é difícil encontrar documentação em língua local.

Para os falantes do português existe o Debian-BR, que tem ampla documentação sobre o Debian: `http://debian-br.cipsga.org.br`.

# Capítulo 2

# Como instalar pacotes

## 2.1 Como instalar pacotes .deb

Para instalar pacotes .deb em sistemas Debian você deve utilizar o sistema `dpkg`. Para isso você tem de estar com o pacote .deb em seu disco e estar logado como root. O seu uso básico é o seguinte:

```
# dpkg -i nomedopacote.deb
```

## 2.2 Como instalar vários pacotes .deb

Se você vai instalar vários pacotes tem de se preocupar com quais pacotes são mais novos, mais antigos, se já estão instalados, etc. É interessante saber das opções a seguir:

-O -> só instala os pacotes previamente selecionados para instalação pelo dselect

-E -> não instala pacotes que tenham a mesma versão do já instalado (se estiver)

-G -> evita "rebaixar" a versão dos pacotes. Se o pacote a ser instalado tiver uma versão menor que o instalado ele não será instalado.

Uma instalação segura de vários .debs pode ser feita assim, por exemplo:

```
# dpkg -iGE *.deb
```

Isso evita ter de usar o dselect para fazer a seleção deles também.

## 2.3 Instalar pacotes com APT

O APT é um sistema que facilita a instalação de pacotes, basta fazer:

```
# apt-get install nomedopacote
```

O APT irá baixar o pacote e suas dependências. Para saber mais sobre ele visite http://debian-br.cipsga.org.br/view.php?doc=apt-howto-pt_BR.

Note que apesar de ter alta qualidade, o APT não suporta todas as possibilidades do .deb, para obter toda a funcionalidade use um "frontend" para ele como o `deity` ou o `aptitude` ou mesmo o `gnome-apt`.

## 2.4 Instalar pacotes com Dselect

O `Dselect` é a ferramenta de manejamento de pacotes padrão do Debian atualmente. Ela é um "frontend" para o APT e para o dpkg.

Para executar o Dselect basta digitar 'dselect' na linha de comando.

Roda-se o Dselect, escolhe-se o método de acesso, seleciona-se "Update" para atualizar a lista de pacotes. Então procede-se a seleção de pacotes.

A primeira tela que aparece depois de se ter selecionado "Select" dá dicas sobre como o Dselect se comporta; presssione a barra de espaço para continuar.

As teclas a seguir podem ser úteis:

+ -> instalar, deixar instalado

- -> desinstalar

_ -> expurgar

H -> manter a versão do pacote selecionado

/ -> procurar uma palavra

\ -> repetir última busca

R -> volta ao estado anterior

Note que as letras em maiúsculas tem de, realmente, ser digitadas em maiúsculas. Para proceder pressione enter. Nesse momento, ou sempre que você mandar instalar um pacote que precise de outros, serão apresentadas as dependências para que você escolha o que fazer.

Outra coisa importante a ser notada é a diferença entre se desinstalar e se expurgar um pacote. Desisntalar é remover o pacote, expurgar é remover o pacote e seus arquivos de configuração.

Depois de selecionados os pacotes, selecione "Install" e saia do Dselect.

Para saber mais do Dselect, veja http://debian-br.cipsga.org.br/view.php?doc=dselect-beginner

## 2.5 Como instalar pacotes sob demanda

Você está compilando aquele programa maravilhoso que você acaba de achar e que resolverá todos os seus problemas e de repente... falta um arquivo. O comportamento padrão nesse caso é bater a cabeça na parede, mas há outras soluções.

O `auto-apt` é uma ferramenta muito importante nessas horas. Ele pode resolver esses problemas parando o programa que precisa do arquivo antes de ele dar erro, pedir para instalar o pacote que provê aquele arquivo com o APT e depois deixar com que o programa continue, sem erros!

Para começar instale o pacote `auto-apt`. O `auto-apt` mantém um banco de dados próprio, por isso, antes de continuar, execute o seguinte:

```
# auto-apt update-local
# auto-apt update
# auto-apt updatedb
```

Estes comandos podem demorar um pouco, mas não precisam ser feitos sempre. Faça, no entanto, com uma certa frequência para manter seu banco de dados completo.

Depois, para fazer com que a mágica aconteça faça:

```
$ auto-apt run comando
```

Troque `comando` pela linha de comando que pode provavelmente precisar de um arquivo. Normalmente usaria-se:

```
$ auto-apt run ./configure
$ auto-apt run make
```

Se você executar `auto-apt run`, somente, o `auto-apt` abre um ambiente dele, no qual qualquer pacote que for necessário será instalado, depois de uma resposta afirmativa à pergunta feita ao usuário.

A pergunta será feita em uma interface gráfica, caso se esteja no X (e tenha os pacotes necessários para tal) ou em texto, caso esteja no console.

## 2.6 Como instalar pacotes .tgz ou .rpm no Debian

Pacotes `.tgz` são pacotes que contêm binários, normalmente utilizados em Slackware. Você pode instalá-los assim:

```
# tar zxpvf arquivo.tgz
```

Estando no diretório `/`. Mas não é boa política já que o sistema de empacotamento não vai tomar conhecimento de que aquele pacote está instalado. O jeito certo é usar o `alien`, assim:

```
$ alien -d arquivo.tgz
```

Isso irá transformar o arquivo `tgz` em um `.deb` que você pode instalar com o `dpkg`.

O mesmo se aplica ao `rpm`, o utilitário `rpm` está disponível no Debian, mas é uma política melhor converter o pacote `rpm` em `deb` para manter a consistência do banco de dados do `dpkg`. O comando do alien é o mesmo para pacotes `rpm`.

## 2.7 Como abrir pacotes .deb com o dpkg-deb

Algumas vezes é necessário se obter os arquivos dentro de um pacote sem instalá-lo, uma situação comum é quando você precisa recuperar o arquivo de configuração original ou algum arquivo de uma versão antiga do pacote. O utilitário usado para isto é o `dpkg-deb`:

```
$ dpkg-deb -x nome_do_pacote.deb /tmp/destino
```

Com o comando acima, o pacote `.deb` terá todos os seus arquivos descompactados para dentro do diretório `/tmp/destino`. A estrutura criada em `/tmp/destino` é a mesma que seria colocada no diretório raíz (`/`).

## 2.8 Como usar pacotes .deb sem o dpkg

Em sistemas não-Debian, que não têm o `dpkg` ou por algum motivo especial, você pode querer obter o conteúdo de um `.deb` sem utitilizar-se do sistema de empacotamento.

Os `.deb`'s foram concebidos para que pudessem ser abertos em qualquer sistema unix, tornando fácil essa tarefa. Para abrir um pacote `.deb` você pode usar:

```
$ ar -x pacote.deb
```

Esse comando extrai três arquivos: `debian-binary`, que indica a versão da especificação do formato .deb que esse pacote segue, `data.tar.gz`, que contém uma árvore de diretórios com os arquivos contidos no pacote e `control.tar.gz`, que contém informações de controle sobre o pacote.

Para "instalar" o programa contido no `deb`, então, basta ir para o diretório raiz ("/") e executar:

```
# tar zxpvf data.tar.gz
```

Provendo o caminho completo para o `data.tar.gz`, caso ele não se encontre no raiz.

# Capítulo 3

# Selecionando Pacotes

## 3.1 Como obter uma lista de pacotes instalados

Muitas vezes quer-se fazer uma seleção rápida de pacotes para outras máquinas recém-instaladas e assim instalar rapidamente várias máquinas. Para obter uma lista das seleções atuais do seu sistema use o comando a seguir:

```
$ dpkg --get-selections
```

Para gravar isso em um arquivo basta usar os recursos de redireção da shell assim:

```
$ dpkg --get-selections > lista-de-pacotes.txt
```

Outra maneira de se obter uma lista de pacotes é executando:

```
$ dpkg -l
```

As linhas que começam com "ii" mostram os pacotes instalados, as linhas que começam com "rc" mostram pacotes que já estiveram instalados, foram removidos mas continuam com seus arquivos de configuração instalados. Para remover de vez os arquivos de configuração use a opção `--purge` ao remover (tanto no APT quanto no dpkg).

## 3.2 Como definir uma lista de pacotes para outra máquina

Depois de conseguir a lista de pacotes com `dpkg --get-selections`, você quer definir aquela mesma seleção de pacotes para outra instalação, basta usar o comando:

```
# dpkg --set-selections < lista-de-pacotes.txt
```

E usar:

```
# dselect install
```

para completar a instalação a partir da mídia disponível (normalmente um CD ou um mirror do Debian, depende das fontes que foram configuradas no APT e para o Dselect.)

Note porém que pacotes que não estavam instalados na máquina original não serão removidos da máquina destino caso estejam instalados na mesma.

## 3.3 Como manter os pacotes numa mesma versão

É possível usar a função `set-selections` com apenas um pacote. Usamos isso para manter um pacote numa mesma versão. Isso é útil em casos em que se quer usar uma versão específica de um programa.

Por exemplo: eu costumo fazer alterações no `gdm` para que ele tenha dois botões a mais e, portanto, não quero que uma atualização seja feita nesse pacote até que eu tenha preparado minhas modificações no próximo. Para fazer isso, basta usar:

```
# echo nomedopacote hold | dpkg --set-selections
```

Isso vai prender o pacote `nomedopacote` na versão atual. Para voltar ao estado normal:

```
# echo nomedopacote install | dpkg --set-selections
```

## 3.4 Como procurar pacotes

Para procurar um pacote para instalação você pode usar o Dselect (veja 'Instalar pacotes com Dselect' on page 4) ou a ferramenta `apt-cache` da seguinte forma:

```
$ apt-cache search palavra-chave
```

Palavra chave pode ser qualquer palavra que tenha a ver com o que você quer. Ela será procurada na descrição do pacote e em seu nome. Você pode usar mais de uma palavra chave para aumentar a especificidade da sua busca.

# Capítulo 4

# Obtendo informações sobre pacotes

## 4.1 Como saber o estado de um pacote

Para saber se um pacote se encontra instalado no sistema e qual sua versão há duas maneiras:

```
$ dpkg -l | grep pacote
```

ou

```
$ dpkg -s pacote
```

O segundo é mais detalhado.

## 4.2 Como obter o nome do pacote que contém um certo arquivo?

Para saber a qual pacote pertence um arquivo, caso este pacote esteja instalado, pode-se usar:

```
$ dpkg -S arquivo
```

*arquivo* pode ser um nome de arquivo normal. O resultado é assim, por exemplo:

```
# dpkg -S bin/ls
lsof: /usr/sbin/lsof
fileutils: /bin/ls
modutils: /sbin/lsmod
e2fsprogs: /usr/bin/lsattr
syslinux: /usr/bin/lss16toppm
gnupg: /usr/bin/lspgpot
sysutils: /usr/bin/lsdev
```

Os pacotes são listados à esquerda e os arquivos que batem com o padrão passado à direita. Outra forma seria:

```
# dpkg -S debiandoc2html
debiandoc-sgml: /usr/share/man/man1/debiandoc2html.1.gz
debiandoc-sgml: /usr/bin/debiandoc2html
```

Se o pacote não está instalado, pode-se usar o `auto-apt`:

```
$ auto-apt check /caminho/arquivo
```

Perceba que o caminho para o arquivo tem de ser absoluto. Um exemplo real:

```
# auto-apt check /usr/bin/zsoelim
doc/man-db
```

Isto significa que o arquivo se encontra no pacote `man-db` que está na seção `doc` do Debian. Uma outra opção útil do auto-apt é:

```
auto-apt list
```

Esta opção lista os nomes de todos os arquivos disponíveis para instalação/instalados e seus respectivos pacotes. Usado em conjunto com o grep é muito útil.

## 4.3 Como ter informações mais detalhadas?

Para ver uma descrição do pacote, o nome do mantenedor que o empacota, versão, dependências, etc, basta usar:

```
$ apt-cache show nomedopacote
```

Para ver a lista de arquivos de um pacote .deb ainda não instalado pode-se usar:

```
$ dpkg -c nomedopacote.deb
```

Para ver a lista de arquivos de um pacote instalado, use:

```
$ dpkg -L nomedopacote
```

Existe ainda o programa `dlocate`. Para usá-lo é necessário instalar o pacote `dlocate` Ele tem vantagens sobre o dpkg. Inúmeras. Além da velocidade superior, ele pode mostrar informações muito mais completas.

As opções `-S`, `-L`, `-s` e `-l` funcionam quase exatamente como as do `dpkg`, veja acima. No entanto, o ganho de velocidade e detalhes é incrível.

Usando:

```
$ dlocate -conf
```

você consegue listar os arquivos que são marcados 'conffiles' do pacote (arquivos de configuração gerenciados pelo dpkg).

Uma outra opção muito útil é a `-man`, que lista as páginas de manual de um pacote:

```
$ dlocate -man gkdial-gnome
1 gkdial
```

## 4.4 Informações ainda mais detalhadas?

O diretório `/var/lib/dpkg/info` guarda os scripts e outros arquivos de controle dos pacotes. Todo arquivo começa com o nome do pacote ao qual pertence. Os scripts que são executados logo após a instalação do pacote (postinst), por exemplo, são encontrados ali.

O arquivo `/var/lib/dpkg/status` guarda o "estado" atual dos pacotes e informações mais detalhadas sobre os que estão instalados. O arquivo `/var/lib/dpkg/available` lista pacotes disponíveis e suas descrições. Esses arquivos podem ser grandes aliados quando se procura informações sobre pacotes.

O arquivo `Contents-???.gz` (onde ??? é a arquitetura na qual você roda Debian) pode ser também uma ótima fonte de informações. Ele se encontra no diretório `/debian/dists/???` onde ??? é a distribuição que você usa (potato, woody, sid ou stable, testing, unstable) e lista todos os arquivos que todos os pacotes contêm.

## 4.5 Documentação dos pacotes

Todo pacote Debian tem um changelog e um copyright. Normalmente juntam-se a esses documentação adicional do programa como READMEs, manuais de usuário e arquivos de créditos. Toda essa informação pode ser encontrada sempre em `/usr/share/doc/pacote`.

Há também documentação em forma de páginas de manual, que pode ser acessada com o comando `man` e em forma de documentos info, que pode ser encontrada com o comando `info` ou com algum outro leitor de info como o `emacs`, entre outros.

É bom também dar uma olhada no programa `gnome-help-browser` que faz parte do GNOME. Ele é uma forma centralizada de obter ajuda no sistema. Basta digitar `man:página` (por exemplo man:ls) para obter uma página html que mostra o conteúdo da página de manual. O mesmo pode ser aplicado para documentação info.

# Capítulo 5

# Como usar pacotes fonte

## 5.1 O que são os pacotes fonte

Pacotes fonte no Debian não são necessariamente um pacote .deb. Ao contrário do src.rpm, o Debian tem três arquivos que representam o pacote fonte. O arquivo `.orig.tar.gz` (ou `.tar.gz` no caso de pacotes nativos Debian), o `.diff.gz` e o `.dsc`.

O `orig.tar.gz` tem o fonte original do programa, sem modificação alguma, o `.diff.gz` guarda as modificações feitas no pacote para "debianizá-lo" e, finalmente, o `.dsc` é um arquivo de controle usado pelo sistema de empacotamento para lidar com o pacote fonte.

## 5.2 Como baixar pacotes fonte com o APT

Pode-se baixar pacotes fonte com o APT. Para isso basta digitar:

```
$ apt-get source pacote
```

Esse comando baixa os três arquivos necessários e cria o diretório `pacote-versão`. Para que o processo de compilação aconteça automaticamente basta executar:

```
$ apt-get -b source pacote
```

Isso irá fazer com que o pacote fonte seja baixado e automaticamente compilado após o download.

## 5.3 Como usar pacotes fonte com dpkg-source

Outra forma de usar pacotes fonte é baixar por si mesmo os três arquivos necessários e então usar as ferramentas providas pelo sistema de gerenciamento de pacotes para usá-los. Para criar o diretório `pacote-versão` debianizado a partir dos três arquivos basta rodar:

```
$ dpkg-source -x pacote.dsc
```

Isso vai abrir o `orig.tar.gz` e aplicar o `.diff.gz`. A partir desse momento pode-se usar o `dpkg-buildpackage -rfakeroot -us -uc` para construir esse pacote.

É o `dpkg-source` que abre o pacote fonte no diretório corrente após um `apt-get source pacote`.

## 5.4 Como compilar um pacote fonte

Para compilar um pacote fonte (torná-lo um .deb) basta entrar no diretório criado pelo `dpkg-source` e digitar:

```
$ dpkg-buildpackage -rfakeroot -b -us -uc
```

Isso construirá no diretório pai (onde está o diretório `pacote-versão`) o pacote .deb daquele programa. Note que o `-rfakeroot` só é necessário caso você não seja `root`.

Você pode também rodar o script `debian/rules` com o argumento `binary`. Dentro do diretório `pacote-versão` faça:

```
$ debian/rules binary
```

Use o `fakeroot` caso não esteja como root, colocando-o no início da linha acima.

## 5.5 Como mudar as opções de compilação de um pacote fonte

Normalmente você baixou o fonte de um determinado pacote para alterar alguma função que vem pre-compilada ou não nele. Para mudar essas opções basta editar, na maioria dos casos, o arquivo `debian/rules`, que fica dentro do diretório aberto pelo `dpkg-source`. Veja a seção 'Como usar pacotes fonte com dpkg-source' on the preceding page para detalhes.

O arquivo `debian/rules` é, na maioria das vezes, um `makefile` que é chamado para construir o pacote. Nele estão as chamadas ao `./configure` e os comandos de compilação/instalação do programa. Alterando-se esses pode-se mudar as características do pacote que é construído.

## 5.6 Como resolver dependências de compilação

Para compilar pacotes, normalmente, você vai precisar de algumas bibliotecas e headers que podem não estar instalados. Essas são as chamadas dependências de compilação. Para resolver, você pode olhar no arquivo `debian/control` que está dentro do diretório `pacote-versão` a linha `Build-Depends:` e instalar os pacotes listados ali.

O APT mais novo, que está atualmente no woody tem uma função especial que é capaz de checar essas dependências automaticamente. Para isso basta executar:

```
# apt-get build-dep pacote
```

O `pacote` da linha de comando é o pacote do qual se quer pegar as dependências de compilação. Note que esse comando não baixa o fonte do `pacote`.

# Capítulo 6

# Criar pacotes Debian

## 6.1 Como construir um pacote deb

O jeito correto de se construir um pacote .deb é ler os manuais de política e entender corretamente como funcionam. Você pode ler o Guia dos Novos Mantenedores Debian (http://debian-br.cipsga.org.br/view.php?doc=maint-guide), mas há muita documentação em http://www.debian.org/devel.

## 6.2 Como construir pacotes para preencher dependências

Muitas vezes você quer instalar um pacote compilado por você mesmo e quer que o sistema se comporte como se soubesse que ele está instalado. Para conseguir isso você pode usar o `equivs`. Não vou documentar aqui como usar o programa, já que ele já está documentado na seção `Ajudantes muito úteis` do Como Usar o APT (http://debian-br.cipsga.org.br/view.php?doc=apt-howto-pt_BR), que pode ser lido na página do Projeto de Documentação Debian: http://www.debian.org/doc/ddp.

## 6.3 Como reconstruir um .deb

Você precisa de um deb para instalar em uma máquina mas não tem o CD, precisa pegar da Internet. Mas... um dos computadores tem esse pacote instalado. Você pode recriar o .deb a partir dos arquivos instalados usando:

```
# dpkg-repack nomedopacote
```

Você precisa estar como root, ou deve usar o programa 'fakeroot' da seguinte forma:

```
$ fakeroot dpkg-repack nomedopacote
```

Note que pode ser que o processo dpkg-repack tenha de usar alguns arquivos de configuração ou qualquer outro tipo de arquivo que necessitam de root para serem lidos e se esse for o caso nem mesmo o fakeroot poderá ajudar, deve-se usá-lo como root nesses casos.

# Capítulo 7

# Como configurar o Debian

## 7.1 O Debconf

O Debconf é o sistema de configuração centralizado do Debian para pacotes que precisam fazer perguntas ao usuário na instalação. Ele funciona baseado em um "backend", que tem um banco de dados com todas as informações já respondidas, e um "frontend", que é a interface que faz as perguntas e mostra os avisos ao usuário.

Toda pergunta respondida fica guardada no banco de dados do Debconf e nunca mais será perguntada, ao menos que seja realmente necessário.

A configuração do sistema base, logo após a instalação, se dá pelo debconf. O próprio debconf se configura por si mesmo.

As perguntas tem níveis de importância e o usuário pode selecionar quais perguntas quer ver ou não (por nível de importância). Às perguntas não mostradas serão considerados os valores padrão.

## 7.2 Como configurar um pacote após a instalação

Muitas vezes você gostaria de responder novamente a algumas perguntas feitas durante a instalação de um pacote. Para isso existe a ferramenta conhecida como `dpkg-reconfigure`.

É importante lembrar que somente pacotes que usem o debconf podem ser reconfigurados com essa ferramenta. Use-a da seguinte forma:

```
# dpkg-reconfigure pacote
```

Eu criei há pouco tempo um programa em Gtk chamado `GKDebconf`. Ele é uma GUI para o `dpkg-reconfigure`. Para usá-lo basta instalar o pacote `gkdebconf`.

# Capítulo 8

# Quando dois pacotes fazem a mesma coisa

## 8.1 O que são alternativas

Muitas vezes dois ou mais pacotes desempenham a mesma função ou fornecem o mesmo programa de um modo diferente mas podem ser instalados ao mesmo tempo, definindo assim alternativas.

Um bom exemplo são os famosos "emuladores de terminal", como o `xterm` ou o `rxvt`. Esses pacotes fornecem o programa `x-terminal-emulator`, assim você pode chamar `x-terminal-emulator` que o pacote que for o dono da alternativa será chamado.

## 8.2 Vendo alternativas

Para ver as alternativas entre as quais você pode escolher para um determinado comando basta rodar:

```
$ update-alternatives --display nome
```

Onde *nome* é o nome da alternativa. Os nomes das alternativas são os links presentes no diretório `/etc/alternatives`

## 8.3 Configurando alternativas

Para escolher entre alternativas você pode usar o seguinte comando:

```
# update-alternatives --config nome
```

Lembrando que o *nome* é o nome de um dos links em `/etc/alternatives`. Por exemplo:

```
# update-alternatives --config x-terminal-emulator

There are 3 programs which provide ‘x-terminal-emulator’.

  Selection    Command
-----------------------------------------------
 +    1        /usr/X11R6/bin/xterm
      2        /usr/X11R6/bin/uxterm
*     3        /usr/bin/gnome-terminal.wrapper

Enter to keep the default[*], or type selection number: 1
Using ‘/usr/X11R6/bin/xterm’ to provide ‘x-terminal-emulator’.
```

Note que a entrada em que está um `'*'` (asterisco) é a padrão, a que está sendo usada. A que está marcada com um `'+'` é a "melhor" entrada, seria a usada no caso de a alternativa ser escolhida automaticamente (veremos adiante).

Então, depois de rodar o que rodei acima, o `xterm` será meu `x-terminal-emulator`.

## 8.4 Alternativas automáticas

Na verdade é assim que as alternativas se comportam caso você não interfira com elas. Mas caso você tenha mexido alguma coisa e queira voltar ao modo automático basta fazer:

```
# update-alternatives --auto nome
```

Com isso a "melhor" entrada (aquela que tem um `'+'` na primeira coluna da `--config` ou que é mostrada como `"'best' version"` pelo `--display`) será selecionada automaticamente sempre.

## 8.5 Lidando com alternativas pelos links

Todo pacote que é parte de uma alternativa, quando se instala pela primeira vez cria um link em `/etc/alternatives` apontando para seu executável. O executável que será o principal das alternativas, por sua vez, é um link para o link que está no diretório citado. Complicado? Nem tanto, veja o exemplo:

```
$ ls -l /usr/bin/x-terminal-emulator
lrwxrwxrwx    1 root     root           37 Jul 16  2000 /usr/bin/x-terminal-e
$ ls -l /etc/alternatives/x-terminal-emulator
lrwxrwxrwx    1 root     root           31 Nov 24 18:00 /etc/alternatives/x-t
```

Dá pra perceber que eu uso o `gnome-terminal` como meu `x-terminal-emulator`, então toda vez que algum programa chama tal programa o que se abre é o `gnome-terminal`. Você pode mudar suas alternativas mudando esses links.

## 8.6 Como criar ou adicionar alternativas

O `update-alternatives` permite que você adicione facilmente uma alternativa ao sistema. Por exemplo, posso criar a alternativa `web-browser`, onde coloco meus navegadores web.

Para isso faço:

```
# update-alternatives --install comando nome comando-real prioridade
```

O que isso faz? Instala uma nova alternativa na qual você pode chamar *comando* para rodar *comando-real*. Um exemplo prático:

```
# update-alternatives --install /usr/bin/web-browser web-browser /usr/bin/moz
# update-alternatives --install /usr/bin/web-browser web-browser /usr/bin/gal
```

Isso cria `/usr/bin/web-browser` que aponta, primeiramente, para o `galeon`, já que esse tem prioridade maior e a alternativa está configurada para `auto`. O link criado em `/etc/alternativos` é o argumento *nome* (veja o primeiro exemplo), que serve para muitas outras funções do `update-alternatives`.

Note, porém, que `web-browser` não tem uma manpage. Seria lógico ligar a ele a manpage do browser que está sendo usado como `web-browser` em determinado momento, certo? Para isso que existe a opção `--slave`. Seria algo assim:

```
# update-alternatives --install /usr/bin/web-browser web-browser /usr/bin/moz
```

Não é necessário explicar os argumentos de `--slave`, são todos os do `--install` menos a prioridade.

## 8.7 Removendo alternativas

Remover alternativas é muito simples. Supondo que quero remover o `galeon` da alternativa `web-browser` que criei na seção anterior basta fazer:

```
# update-alternatives --remove web-browser /usr/bin/galeon
```

Ou seja:

```
# update-alternatives --remove nome comando-real
```

# Capítulo 9

# Usando o Sistema de Menus do Debian

## 9.1 Instalando o Sistema de Menus (e o que é)

O Sistema de Menus do Debian é único, centralizado e muito bem projetado. Nenhum outro sistema tem um sistema de menus tão consistente.

Esse sistema cria, para cada gerenciador de janelas instalado, um item de menu para cada programa instalado no sistema. Isso torna fácil a localização dos programas instalados no sistema e evita ter de adicionar um item de menu para cada gerenciador de janelas dentro do pacote de cada programa.

Para usar esse sistema certifique-se de instalar o pacote `menu`. Veja 'Como instalar pacotes' on page 3 para saber como fazer isso.

Para atualizar os menus de todos os gerenciadores de janelas instalados basta rodar o `update-menus`, mas isso já é feito automáticamente pelos programas que usam o sistema.

Dentro de todo gerenciador de janela será criado um submenu dentro do menu principal chamado `Debian`, nele estarão os itens de menu dos programas.

## 9.2 Como criar novas entradas globais no Sistema de Menus

Um administrador de sistemas pode querer colocar novos itens no Sistema de Menus para que todo usuário possa ver esse item. Para isso basta colocar uma entrada de menu no diretório `/etc/menu`. Exemplos de entradas de menu podem ser encontradas em `/usr/lib/menu`, veja 'Um exemplo de entrada de menu' on the following page para um exemplo de como fazer.

## 9.3 Como modificar uma entrada globalmente

Para passar por cima de uma definição de item de menu basta copiar o arquivo de `/usr/lib/menu` para `/etc/menu` e modificar as entradas. Para saber como modificar um arquivo do sistema

de menu veja 'Um exemplo de entrada de menu' on the current page.

## 9.4 Como remover uma entrada globalmente

Para remover uma entrada de menu globalmente basta criar um arquivo branco no `/etc/menu` com o mesmo nome do arquivo que provê aquela entrada no `/usr/lib/menu`. Isso pode ser feito com `touch /etc/menu/programa`.

## 9.5 Como lidar com menus se você não é root

Os usuários do sistema têm também as mesmas possibilidades que o root, mas não fazem as mudanças globalmente e sim para seus próprios menus.

Ao invés de usar o diretório `/etc/menu`, os usuários normais usam o diretório `.menu` dentro do seu próprio home. Por exemplo, o usuário `kov` usaria o diretório `/home/kov/.menu`.

Veja as seções anteriores para saber como proceder para incluir, editar e remover entradas.

## 9.6 Um exemplo de entrada de menu

Uma entrada de menu seria como segue:

```
?package(gkdial):\
      needs="x11"\
      hints="Discador feito em Gtk" \
      section="Apps/Net"\
      title="GkDial"\
      command="/usr/bin/gkdial"\
      icon="/usr/share/pixmaps/gkdial.xpm"
```

Dentro do parênteses de ?package deve-se colocar o nome do pacote que provê esse programa, assim a entrada de menu só será mostrada caso o pacote esteja instalado.

Se você quiser usar um programa externo, que não esteja instalado como .deb ou quer mostrar uma entrada de menu mesmo sem o pacote instalado, basta colocar:

```
?package(local.programa):\
```

O parâmetro `needs` diz ao sistema de menu o que o programa precisa pra rodar. Pode ser `x11`, para programas que rodam sob o X; `text`, para programas que precisam de um terminal como o xterm; `vc`, para programas que só rodam em console e `wm` para programas que precisam de um gerenciador de janelas específico.

O `hints` é usado para dicas, aquelas mensagens que aparecem quando se põe o mouse em cima de um item de menu explicando sua função.

O `section` indica em qual submenu, dentro do menu do Debian deve entrar esse item, as entradas válidas estão listadas em `/usr/share/doc/menu/html/ch3.html`.

O `title` configura o nome que o item de menu terá. O `command` define a linha de comando a ser executada (caminho absoluto) e o `icon` define qual ícone será utilizado para a entrada de menu.

# Capítulo 10

# Como lidar com o Kernel no Debian

## 10.1 Como atualizar o Kernel no Debian

O kernel é o núcleo do sistema. É o que conhecemos como `Linux`. O Debian pretende ser um sistema independente de kernel, portanto você pode usar o Debian GNU/Hurd ao invés do Debian GNU/Linux e logo será possível usar Debian GNU/BSD.

Para atualizar kernel no Debian basta usar o `apt-cache` para descobrir quais versões estão disponíveis e então usar o `apt-get` para baixar a nova versão. Os pacotes de kernel se chamam `kernel-image-versão` onde `versão` é a versão que se quer usar.

Veja 'Como procurar pacotes' on page 8 e 'Instalar pacotes com APT' on page 3 para saber como fazer isso.

## 10.2 Como atualizar para o Linux 2.4

O kernel 2.4 não é suportado oficialmente para a versão 2.2 do Debian. A versão 3.0 virá com ele como opção. No entanto, Adrian Bunk colocou à disposição um set de pacotes disponíveis para quem quer atualizar o kernel para 2.4.

As linhas APT para ele é:

```
deb http://www.fs.tum.de/~bunk/debian potato main
deb-src http://www.fs.tum.de/~bunk/debian potato main
```

Note que é importante uma atualização de vários pacotes antes de se atualizar o kernel para que ele funcione corretamente, não force uma situação diversa disso.

## 10.3 Como compilar kernel no Debian

Todos já estão acostumados àquela imensa lista de comandos para se compilar kernel (bem, nem todos). No Debian a coisa é bem mais simples.

Para começar veja 'Como instalar pacotes' on page 3 e instale o pacote `kernel-package`. Você precisa também, claro, do fonte do kernel. Você pode baixá-lo a partir de `http://www.kernel.org` ou instalar o pacote `kernel-source-versão` correspondente. Nesse último caso, um arquivo será instalado em `/usr/src` com nome `kernel-source-versão.tar.bz2` descompacte-o e crie um link do diretório recém criado para `/usr/src/linux`, dessa forma:

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

Depois basta fazer, dentro do diretório principal do kernel – `/usr/src/linux`.:

```
$ make menuconfig
$ make-kpkg clean
$ make-kpkg kernel_image
```

Você pode substituir `make menuconfig` por `make xconfig` para fazer a configuração pelo X. Se esses comandos falharem, você pode ter de instalar o pacote `libncurses5-dev` (ou `libncurses4-dev` em sistemas mais antigos).

Esses comandos criarão um .deb do Kernel recém compilado. Basta instalar como descrito em 'Como instalar pacotes .deb' on page 3.

Você também pode definir 'revisões' para cada kernel que construir. Para isso use `--revision`, por exemplo:

```
$ make-kpkg --revision kov1.0 kernel_image
```

Note que é necessário que haja dígitos na sua 'revisão'. Depois do kernel ser instalado ele perguntará se deseja criar um disco de boot e se deseja rodar o lilo. Normalmente é uma boa idéia conferir o /etc/lilo.conf antes de rodar o lilo novamente, mas normalmente não haverá problemas em deixar que o processo de construção o faça, já que os links `vmlinuz` e `vmlinuz.old` são recriados corretamente durante a instalação do deb do kernel.

## 10.4 Como configurar os módulos do kernel com o modconf

No Debian, para lidar com módulos de kernel existe uma ferramenta chamada `modconf`. Ela é a ferramenta usada para configurar os módulos durante a instalação e pode ser chamada depois, como root.

Logo depois de entrar no modconf, uma lista de seções será listada. Dentro das sessões são listados os vários módulos que podem ser instalados (ou seja, carregados) no sistema.

Para carregar um módulo basta dar enter sobre ele. Alguns módulos requerem opções adicionais para serem carregados como por exemplo as placas NE 2000 ISA. Logo que você digita enter, o modconf perguntará por essas opções. Para a placa NE 2000 do meu 486, por exemplo eu coloco:

```
io=0x300 irq=5
```

Essas opções e os módulos carregados serão guardados para toda vez que se iniciar a máquina ter-se a mesma configuração.

## 10.5 Como configurar módulos na mão

Os módulos que serão carregados na inicialização do sistema podem ser configurados também sem o modconf. Basta listá-los um por linha no arquivo `/etc/modules`.

Só há um problema. Alguns módulos, como o da placa NE 2000 precisam de opções para serem carregados. Essas configurações, num sistema comum seriam configuradas em `/etc/modules.conf`. Mas, o pacote modutils do Debian sempre sobrescreve esse arquivo porque o gera automaticamente, a partir de vários outros.

O certo é criar um arquivo em `/etc/modutils/` e rodar o programa `update-modules`. Por exemplo. Para minha placa de som Opti931 (clone de Sound Blaster Pro), que usa o módulo mad16.o, eu uso o seguinte, no arquivo `/etc/modutils/mad16`:

```
alias mixer0 mad16
alias audio0 mad16
alias midi0  mad16
alias synth0 opl3
options sb mad16=1
options mad16 irq=10 dma=0 dma16=1 io=0x530 joystick=1 cdtype=0
options opl3 io=0x388
post-install mad16 /sbin/adreroute 14 8 15 3 16 6
```

E coloco "`mad16`" no `/etc/modules`. Mais informações nas manpages `insmod(8)`,`modprobe(8)`, `modules(5)` e `modules.conf(5)`.

# Capítulo 11

# Como lidar com os scripts de inicialização

## 11.1 Entendendo os scripts de inicialização

O primeiro processo rodado pelo sistema após o boot do kernel chama-se `init`. Ele tem como tarefa, além de outras coisas, checar a integridade das partições e rodar os serviços da máquina.

Existem dois estilos de init: o BSD e o SYS V. Cada um tem suas peculiaridades. O Debian usa init SYS V. Os scripts são armazenados em `/etc/init.d` e links são criados em `/etc/rc?.d`, onde o ? representa o runlevel. Existe ainda o diretório `/etc/rc.boot`, rodado antes de todos os runlevels mas só durante o boot. Esse diretório já é obsoleto e não deve ser usado.

Os links que se encontram nos diretórios `/etc/rc?.d` estão no seguinte formato:

```
SNNnome
KNNnome
```

Onde NN é um número qualquer. Os que começam com S serão iniciados e os que começam com K serão matados. O número, representado pelo NN define a ordem. Começa do menor e vai para o maior.

Todo script do `/etc/init.d` (e consequentemente dos `/etc/rc?.d`) suportam os argumentos “start”, “stop” e “restart” que podem ser usados para iniciar, parar e reiniciar os serviços, respectivamente. Para reiniciar o servidor apache, por exemplo, executa-se:

```
# /etc/init.d/apache restart
```

Alguns dos scripts suportam também “reload” e/ou “force-reload”. Chamar os scripts com `--help` lista os argumentos suportados.

## 11.2 Gerenciando scripts de inicialização com update-rc.d

O `update-rc.d` é uma ferramenta de gerenciamento de scripts de inicialização. Para remover todos os links de um script, por exemplo faz-se:

```
# update-rc.d -f nome remove
```

Onde `nome` é o nome do script em `/etc/init.d`. Isso é necessário para se criar uma configuração diferente da que foi feita anteriormente.

Se você quer reconfigurar um script de inicialização ou acaba de incluir um script seu em `/etc/init.d` e quer iniciá-lo, basta usar:

```
# update-rc.d nome start NN runlevel . (...)
```

Por exemplo: vamos imaginar que acaba de ser colocado o script maintaince.sh no `/etc/init.d`. Você quer que ele seja iniciado (argumento "start" passado) no runlevel de boot padrão (2) e fechado (argumento "stop") quando se desliga ou reinicia (runlevels 0 e 6) e quer que ele seja um dos últimos a ser iniciado/parado. Faz-se então:

```
# update-rc.d maintaince.sh start 97 2 . stop 97 0 . stop 97 6 .
```

Como se pode ver, as configurações são separadas por um ponto (.) e há um também no final da linha de comando. Para somente testar o que será feito adicione um "-n" depois do nome do comando.

Para facilitar as coisas, pode-se usar o update-rc.d com as configurações padrão. Ou seja: iniciar nos runlevels 2, 3, 4 e 5 e finalizar nos 0, 1 e 6. Basta usar:

```
# update-rc.d nome defaults
```

O `update-rc.d` assume a ordem de execução como 20, isso pode ser trocado passando o número logo após a palavra `defaults`. Para usar ordem de execução diferente para iniciar e desligar basta passar dois números após a palavra `defaults` – o primeiro será a ordem de início e o segundo de finalização do script.

## 11.3 Alternativa ao sistema rc.d padrão de links: file-rc

O esquema tradicional do SYS V para implementar os runlevels através de links nos diretórios `/etc/rc?.d` traz o problema de não conseguir fazer a inspeção da sua configuração de runlevels facilmente. Felizmente o Debian nos dá uma alternativa muito boa: **file-rc**.

Para instalá-lo, basta um

```
# apt-get install file-rc
```

Basicamente, ele substitui aqueles links intermináveis e de difícil manutenção por um arquivo único: `/etc/runlevel.conf`. Aqui está um segmento de um `/etc/runlevel.conf` típico:

```
#/etc/runlevel.conf
#
#Formato:
#<ordem> <desligado>   <ligado>        <script>
05       -            0               /etc/init.d/halt
05       -            1               /etc/init.d/single
05       -            6               /etc/init.d/reboot
10       0,1,6        2,3,4,5         /etc/init.d/sysklogd
12       0,1,6        2,3,4,5         /etc/init.d/kerneld
[..]
89       0,1,6        2,3,4,5         /etc/init.d/cron
99       -            2,3,4,5         /etc/init.d/rmnologin
99       0,1,6        2,3,4,5         /etc/init.d/xdm
```

A última coluna é o script que será executado. A primeira coluna é a ordem em que o script será executado no boot. Isso é equivalente ao **NN** dos links S**NN**nome e K**NN**nome vistos acima. A segunda coluna é uma lista separada por vírgulas dos runlevels em que o script será morto (equivalente ao **K** dos links **K**NNnome). A terceira coluna é o mesmo da segunda, mas para os runlevels em que o script será executado (equivalente ao **S** dos links **S**NNnome).

Dessa forma, verifique a linha com o número 10:

```
10      0,1,6   2,3,4,5         /etc/init.d/sysklogd
```

Isso significa que o script `/etc/init.d/sysklogd` será executado nos runlevels 2, 3, 4 e 5, e será morto nos runlevels 0, 1 e 6. Além disso, pelo número de ordem podemos dizer que ele será avaliado (para execução ou morte) imediatamente antes do script `/etc/init.d/kerneld` e imediatamente depois de `/etc/init.d/reboot`.

Como último exemplo, verifique a linha número 99:

```
99      -       2,3,4,5         /etc/init.d/rmnologin
```

Isso significa que o script `/etc/init.d/rmnologin` **não** será morto nunca naquele número de ordem (é isso que significa o "**-**"), e será executado nos runlevels 2, 3, 4 e 5. Se o hífen for utilizado na terceira coluna, isso significa que o script nunca será executado naquele número de ordem.

Adicionalmente, um programa **update-rc.d** é fornecido no pacote para que a administração do boot tipo SYS V seja feita da mesma forma (veja a seção anterior), muito embora eu duvide que, depois de se familiarizar com o `/etc/runlevel.conf`, você ainda vai usar o `update-rc.d`.

# Capítulo 12

# Agradecimentos

Um grande obrigado ao Projeto Debian e a todos os seus desenvolvedores por tornarem real um sistema operacional que se aproxima da perfeição.

Ao Projeto Debian-BR por tornar possível usar esse sistema em português.

Em especial, agradecimentos para:

**Pablo Lorenzoni** (spectra) <spectra@debian.org> pela seção sobre o file-rc

**Gleydson Maziolli da Silva** (Gleydson) <gleydson@debian.org> pela seção sobre o dpkg-deb

**Luis Alberto Garcia Cipriano** <laciprianо@uol.com.br> pelas sugestões valiosas

**Gustavo De Nardin** (spuk) <nardin@inf.ufsc.br> pelas dicas sobre o sistema de menu

**Eduardo Ochs** (edrx) <edrx@inx.com.br> http://angg.twu.net pelas valorosas sugestões de inclusão de informações

**Fabio Grezele** <fg@anhembi.br> pelas correções gramaticais

**Lucas Rocha** <lucasr@led.ufba.br> pela colaboração na melhoria da seção sobre pacotes fonte.

**Felipe Fernandes** <lepow@terra.com.br> pelas dicas para melhoria da seção sobre compilação de kernel.