Como usar o APT

Gustavo Noronha Silva <kov@debian.org>

1.8.11 - Agosto de 2005

Resumo

Esse documento pretende levar ao usuário um bom conhecimento das funções do utilitário de empacotamento do Debian, APT. É objetivo real deste, facilitar a vida dos novos usuários de Debian ou ajudar aqueles que desejam se aprofundar no conhecimento da administração desse sistema. Ele foi criado para o projeto Debian-BR para melhorar ainda mais o suporte à distribuição Debian oferecido ao usuário falante da língua portuguesa.

Nota de Copyright

Sumário

1 Introdução **1**

2 Configurações Básicas **3**

2.1 O arquivo /etc/apt/sources.list 3

2.2 Como usar o APT localmente 4

2.3 Decidindo qual mirror incluir no sources.list: netselect, netselect-apt 5

2.4 Colocando um CDROM na sources.list 6

3 Gerenciando pacotes **7**

3.1 Atualizando a lista de pacotes disponíveis 7

3.2 Instalando pacotes 7

3.3 Removendo pacotes 9

3.4 Atualizando os pacotes 10

3.5 Atualizando para uma nova distribuição 11

3.6 Removendo pacotes que não serão mais usados: apt-get clean e autoclean 13

3.7 Usando em conjunto com o Dselect 14

3.8 Como manter um sistema misto 15

3.9 Como atualizar os pacotes de versões específicas do Debian 16

3.10 Como manter versões específicas de pacotes instaladas (complexo) 16

4 Ajudantes muito úteis **19**

4.1 Como instalar programas compilados localmente: equivs 19

4.2 Removendo arquivos de locale não usados: localepurge 21

4.3 Como saber quais pacotes podem ser atualizados 21

5 Obtendo informações sobre os pacotes. 23

5.1 Descobrindo nome dos pacotes 23

5.2 Usando o dpkg para achar nomes de pacotes 25

5.3 Como instalar pacotes “on demand” 26

5.4 Como descobrir a qual pacote um arquivo pertence 27

5.5 Como manter-se informado das mudanças nos pacotes. 27

6 Lidando com pacotes fonte 29

6.1 Baixando pacotes fonte 29

6.2 Pacotes necessários para compilação de um pacote fonte 30

7 Como lidar com erros? 31

7.1 Erros comuns 31

7.2 Onde consigo ajuda? 32

8 Quais são as distribuições que suportam o APT? 33

9 Agradecimentos 35

10 Novas versões desse tutorial 37

Capítulo 1

Introdução

No início havia o .tar.gz. Os usuários tinham de penar para compilar cada programa usado em seu sistema GNU/Linux, ou outro qualquer. Quando o Debian foi criado, sentiu-se a necessidade de um sistema de gerenciamento de pacotes instalados no sistema. Deu-se a esse sistema o nome de `dpkg`. Assim surgiu o famoso 'pacote'. Logo após a Red Hat resolveu criar seu conhecido sistema rpm.

Rapidamente outro dilema tomou conta das mentes dos produtores de GNU/Linux. Uma maneira rápida, prática e eficiente de se instalar pacotes, gerenciando suas dependências automaticamente e tomando conta de seus arquivos de configuração ao atualizar. Assim, o Debian, novamente pioneiro, criou o APT ou Advanced Packaging Tool, hoje portado pela Conectiva e incorporado por algumas outras distribuições.

Este manual não tenta entrar na área do apt-rpm, como ficou conhecido o APT portado pela Conectiva, mas "patches" são bem vindos para atingir esse objetivo.

Esse manual é baseado na próxima versão do Debian, a `Sarge`.

Capítulo 2

Configurações Básicas

2.1 O arquivo /etc/apt/sources.list

Para seu funcionamento, o APT utiliza-se de um arquivo que lista as 'fontes' de onde ele obterá os pacotes. Esse arquivo é o `/etc/apt/sources.list`.

As entradas desse arquivo são normalmente formadas assim:

```
deb http://host/debian distribuição seção1 seção2 seção3
deb-src http://host/debian distribuição seção1 seção2 seção3
```

É lógico que essas entradas são fictícias e não devem ser usadas. A primeira palavra das linhas, `deb` e `deb-src` indicam o tipo de repositório: se guarda pacotes binários (`deb`), ou seja, os pré-compilados que normalmente usamos ou se guarda pacotes fonte (`deb-src`), que são o fonte original do programa mais o arquivo de controle Debian (`.dsc`) e o `diff.gz` contendo as modificações necessárias para se debianizar o programa.

Normalmente encontramos nos sources.list padrões do Debian o seguinte:

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-fre
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib no
```

Basicamente, isso é o necessário. A primeira linha com `deb` aponta para o repositório oficial, a segunda para o non-US e a terceira para as atualizações de segurança.

As duas últimas linhas estão comentadas, (têm um '#' no início) o que faz com que o apt-get as ignore. Elas são do estilo `deb-src`, ou seja, têm os pacotes fonte do Debian. Se você costuma baixar fontes de programas para testar ou recompilar, descomente-as.

O arquivo `/etc/apt/sources.list` pode conter vários tipos de linhas. O APT sabe lidar com repositórios `http`, `ftp`, `file` (arquivos locais, por exemplo, um diretório que contenha uma ISO montada) e `ssh`, até onde eu sei.

Não se esqueça de rodar `apt-get update` depois de modificar o arquivo `/etc/apt/sources.list`. Você precisa fazer isso para que o APT obtenha as listas de pacotes das fontes que você especificou.

2.2 Como usar o APT localmente

Algumas vezes você tem um monte de pacotes .deb que gostaria de usar o APT para instalar, para que as dependências fossem automaticamente resolvidas.

Para isso crie um diretório e nele coloque os .deb que você quer ter indexados. Por exemplo:

```
# mkdir /root/debs
```

Você pode modificar as definições do arquivo de controle dos pacotes diretamente no repositório usando um arquivo `override`. Dentro desse arquivo você pode definir algumas opções para se sobreporem ao que vem junto do pacote. Ele tem a seguinte forma:

```
pacote prioridade seção
```

Pacote é o nome do pacote, prioridade é low, medium ou high e seção é a seção a qual ele pertence. O nome do arquivo não importa, ele vai ser passado como argumento para o comando `dpkg-scanpackages` mais tarde. Se você não quer usar um arquivo `override` use `/dev/null`.

Ainda no diretório /root faça:

```
# dpkg-scanpackages debs arquivo | gzip > debs/Packages.gz
```

Na linha acima, *arquivo* é o arquivo `override`, o comando gera um arquivo `Packages.gz` que contém informações diversas sobre os pacotes que serão usadas pelo APT. Para usar os pacotes, finalmente, adicione:

```
deb file:/root debs/
```

Depois é só usar os comandos do APT normalmente. Você também pode gerar um repositório de fontes. Para isso use o mesmo procedimento, mas leve em conta que precisa ter os arquivos `.orig.tar.gz`, `.dsc` e `.diff.gz` no diretório e use `Sources.gz` ao invés de `Packages.gz`. O programa a ser usado também difere. É o `dpkg-scansources`. Fica assim:

```
# dpkg-scansources debs | gzip > debs/Sources.gz
```

Note que o `dpkg-scansources` não precisa de um arquivo `override`. A linha pro sources.list fica:

```
deb-src file:/root debs/
```

2.3 Decidindo qual mirror incluir no sources.list: netselect, netselect-apt

Uma dúvida muito freqüente, principalmente dos usuários mais novos, é "qual mirror do Debian colocar no `sources.list`". Para decidir qual o mirror existem várias maneiras. Os mais experientes provavelmente já terão um script pronto medindo os tempos de ping entre os diversos mirrors. Mas já existe um programa que faz isso para nós: **netselect**.

Para instalar o netselect, como sempre:

```
# apt-get install netselect
```

Executando-o sem parâmetros mostra a ajuda. Executando-o com uma lista separada por espaços de hosts (mirrors), ele retornará um escore e um dos hosts. Esse escore leva em consideração o tempo estimado de ping e o número de hops (hosts pelos quais uma requisição de rede deve passar para chegar no host destino), e é inversamente proporcional à velocidade estimada de download (ou seja, quanto menor, melhor). O host retornado é o que obteve o menor escore (a lista dos escores pode ser obtida acrescentanto a opção -vv). Veja o seguinte exemplo:

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes
  365 ftp.debian.org.br
#
```

Isso significa que, dos mirrors incluídos como parâmetros do netselect, `ftp.debian.org.br` foi o melhor, com um escore de 365. (Atenção!! Como isso foi feito do meu computador e a topografia da rede é extremamente diferente dependendo do ponto de contato, esse valor não necessariamente reflete corretamente a velocidade em outros computadores).

Agora, basta colocar o mirror mais rápido encontrado pelo netselect no arquivo `/etc/apt/sources.list` (veja 'O arquivo /etc/apt/sources.list' on page 3) e seguir as dicas em 'Gerenciando pacotes' on page 7.

Observação: uma lista de mirrors pode sempre ser encontrada no arquivo `http://www.debian.org/mirror/mirrors_full`.

A partir da versão 0.3.ds1 o pacote fonte netselect inclui o pacote binário **netselect-apt**, que automatiza o processo descrito acima. Basta usar como parâmetro do script a distribuição (se

não for mencionada, stable é adotada como padrão) e o arquivo `/etc/apt/sources.list` é gerado com os melhores mirrors da seção main e non-US e gravado no diretório atual. O exemplo a seguir gera um sources.list da distribuição stable.

```
 $ ls sources.list
ls: sources.list: File or directory not found
# netselect-apt stable
(...)
# ls -l sources.list
sources.list
#
```

Lembre-se: o arquivo `sources.list` gerado no diretório atual deve ser movido para o diretório `/etc/apt`.

Depois basta seguir as dicas em 'Gerenciando pacotes' on the facing page.

2.4 Colocando um CDROM na sources.list

Se você preferir usar um CDROM para instalar seus pacotes ou atualizar seu sistema automaticamente com o APT, você pode colocá-lo na sua `sources.list`. Para fazer isso, rode o `apt-cdrom` assim:

```
# apt-cdrom add
```

com o CDROM do Debian no drive. Ele irá montar o CDROM, caso seja o certo e irá buscar as informações de pacote dele. Caso sua configuração de CDROM esteja um pouco confusa, você pode usar as seguintes opções:

```
-h           - ajuda do programa
-d diretório - ponto de montagem do CDROM
-r           - renomear um CDROM reconhecido
-m           - não montar
-f           - modo rápido (não atualiza lista de pacotes)
-a           - scan minucioso
```

Por exemplo:

```
# apt-cdrom -d /home/kov/meucdrom add
```

Você ainda pode identificar o CDROM apenas, sem adicioná-lo:

```
# apt-cdrom ident
```

Note que esse programa só funciona caso seu CDROM esteja bem configurado em seu `/etc/fstab`.

Capítulo 3

Gerenciando pacotes

3.1 Atualizando a lista de pacotes disponíveis

O sistema de empacotamento usa um banco de dados próprio para saber quais pacotes estão instalados, quais não estão e quais estão disponíveis para instalação. O apt-get usa esse banco de dados para saber instalar os pacotes solicitados pelo usuário e para saber quais pacotes são necessários para que o pacote selecionado rode perfeitamente.

Para atualizar essa lista, você usa o comando `apt-get update`. Ele procura pelas listas de pacotes nos repositórios indicados no seu arquivo `/etc/apt/sources.list`, veja 'O arquivo /etc/apt/sources.list' on page 3 para maiores informações sobre esse arquivo.

É útil executar esse comando regularmente para saber de possíveis atualizações nos pacotes, principalmente de segurança.

3.2 Instalando pacotes

Finalmente um dos processos mais esperados! Com sua sources.list preparada e a lista de pacotes disponíveis, basta rodar o `apt-get` para ter seu pacote instalado. Por exemplo você pode fazer:

```
# apt-get install xchat
```

O APT vai buscar então em seu banco de dados a versão mais nova desse pacote e vai baixá-la do repositório correspondente na `sources.list`. Caso esse pacote dependa de algum outro – o que é o caso aqui – o APT irá conferir as dependências e instalar os pacotes necessários. Veja esse exemplo:

```
# apt-get install nautilus
Reading Package Lists... Done
```

```
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

O pacote `nautilus` depende das bibliotecas compartilhadas citadas, portanto, o APT as vai buscar no repositório. Se você tivesse especificado os nomes dessas bibliotecas na linha de comando do `apt-get`, o APT não teria perguntado se devia continuar ou não, ele tomaria como certo o seu desejo em instalar todos aqueles pacotes.

Isso significa que o APT só pede confirmação quando precisar instalar pacotes além daqueles que foram solicitados para suprir uma dependência.

As seguintes opções podem ser de utilidade:

```
-h - ajuda
-d - baixar arquivos apenas, não instalar
-f - conserta erros de instalações de pacotes
-s - não agir, apenas simular operação
-y - assume 'sim' para todas as perguntas
-u - mostrar pacotes que serão atualizados também
```

Múltiplos pacotes podem ser solicitados em uma única linha de comando. Os arquivos baixados da rede são colocados no diretório `/var/cache/apt/archives` para serem instalados depois.

Você pode especificar pacotes para remoção na mesma linha de comando. Para isso, basta colocar um '-' logo depois do nome do pacote a ser removido, assim:

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Veja a seção 'Removendo pacotes' on this page para maiores detalhes sobre a remoção de pacotes.

Caso você de alguma forma danifique a instalação de um pacote, ou simplesmente deseja que os arquivos do pacote sejam repostos com a versão mais nova que estiver disponível, você pode usar a opção `--reinstall` assim:

```
# apt-get --reinstall install gdm
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1  not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 Removendo pacotes

Caso você não esteja mais querendo usar um pacote, você pode removê-lo do seu sistema usando o APT. Para isso basta usar: `apt-get remove pacote`. Por exemplo:

```
# apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Como você pode observar no exemplo acima, o APT cuida também de retirar os pacotes que dependem do pacote sendo removido. Não há como remover pacotes sem remover os que são dependentes dele.

Rodando o `apt-get` como acima vai levar à remoção dos pacotes, mas seus arquivos de configuração, caso existam, permanecerão intactos. Para uma remoção completa, faça:

```
# apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Note os '*' na frente dos nomes. Eles indicam que os arquivos de configuração serão removidos.

Assim como no caso do método `install`, você pode usar um sinal gráfico para fazer o processo inverso. No caso da remoção, se você adicionar um sinal `'+'` logo depois do nome do pacote, ao invés de removido ele será instalado. Exemplo:

```
# apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Note que o `apt-get` informa os pacotes extras que serão instalados, ou seja, os pacotes cuja instalação será necessária ao funcionamento do pacote cuja instalação foi solicitada, os que serão removidos e os que serão instalados (incluindo novamente os extras).

3.4 Atualizando os pacotes

A atualização de pacotes é um grande trunfo do sistema APT. Ela é feita com um simples comando: `apt-get upgrade`. Você pode atualizar tanto pacotes dentro de uma mesma distribuição quanto atualizar para uma nova distribuição, mas, para essa última, o comando `apt-get dist-upgrade` é melhor, consulte a seção 'Atualizando para uma nova distribuição' on the next page para maiores detalhes.

É útil usar sempre a opção `-u` para esse comando. Essa opção faz com que o APT mostre os pacotes que serão atualizados. Sem ela você vai fazer uma atualização no escuro. O APT irá baixar as últimas versões de cada pacote e as instalará numa ordem coerente. É sempre importante rodar o `apt-get update` antes. Veja a seção 'Atualizando a lista de pacotes disponíveis' on page 7. Veja esse exemplo:

```
# apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
```

```
    adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
    ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
    libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
    libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

Esse processo é muito simples. Repare que logo no início, o `apt-get` diz que alguns pacotes foram `kept back`. Isso significa que eles têm novas versões que não serão instaladas por algum motivo. Alguns deles são: dependências quebradas (um pacote do qual ele depende não tem uma versão disponível para ser baixada), novas dependências (o pacote passou a depender de novos pacotes desde a última versão).

O primeiro motivo não tem solução limpa, para o segundo basta rodar um `apt-get install` específico para o pacote, que baixará suas dependências. Outra solução, ainda mais limpa, é usar o `dist-upgrade`. Veja seção 'Atualizando para uma nova distribuição' on the current page.

3.5 Atualizando para uma nova distribuição

Essa característica do APT serve para atualizar uma distribuição inteira de uma única tacada, através da internet ou de um novo CD adquirido, ou uma ISO baixada.

Ela é usada também quando mudanças são feitas na interrelação de pacotes já instalados que devem ser atualizados mas são mantidos inalterados (`kept back`).

Por exemplo, supondo que você está usando a versão estável do Debian revisão 0 e compre o CD com a revisão 3, você pode usar o APT para atualizar seu sistema a partir desse novo CD. Para isso use o `apt-cdrom` (veja seção 'Colocando um CDROM na sources.list' on page 6) para adicionar o CD ao seu arquivo `/etc/apt/sources.list` e rode o `apt-get dist-upgrade`.

É importante notar que o APT sempre busca as versões mais novas dos pacotes. Portanto, se seu arquivo `/etc/apt/sources.list` estiver listando um repositório que tenha uma versão mais nova de determinado pacote que contém o CD, ele tenderá a buscá-lo de lá.

No exemplo mostrado na seção 'Atualizando os pacotes' on the preceding page, vimos que alguns pacotes ficaram "`kept back`", vamos solucionar isso agora, com o método `dist-upgrade`:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
```

```
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

Note agora que os pacotes serão atualizados, mas novos pacotes serão instalados (as novas dependências dos pacotes). E que o lilo continua `kept back`. Ele provavelmente tem algum problema mais sério, que não uma nova dependência. Podemos conferir isso rodando:

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

Como notamos na saída acima, o lilo tinha um novo conflito com o pacote `debconf-tiny`, ou seja, não podia ser instalado (isso também implica na atualização) sem que o debconf-tiny fosse removido.

Uma maneira de saber especificamente o que leva um pacote a ser mantido ou removido é:

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
```

```
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

Assim fica fácil perceber que o pacote python1.5-dev não pode ser instalado por culpa de uma dependência não satisfeita: python1.5

3.6 Removendo pacotes que não serão mais usados: apt-get clean e autoclean

Quando você instala um pacote o APT busca das fontes listadas em /etc/apt/sources.list os arquivos necessários e os guarda em um repositório local (`/var/cache/apt/archives/`), e então faz a instalação, veja 'Instalando pacotes' on page 7.

Em algum tempo o repositório local pode crescer e ocupar muito espaço em disco. Felizmente o APT fornece ferramentas para lidar com seu repositório local: os métodos `clean` e `autoclean` do `apt-get`.

O `apt-get clean` remove tudo exceto os arquivos de lock dos diretórios `/var/cache/apt/archives/` e `/var/cache/apt/archives/partial/`. Assim, se você precisar reinstalar um pacote o APT irá buscá-lo novamente.

O `apt-get autoclean` remove apenas os arquivos de pacotes que não possam mais ser baixados.

O exemplo a seguir mostra como o apt-get autoclean funciona:

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

Em /var/cache/apt/archives há dois arquivos para o pacote `logrotate` e um para o pacote `gpm`.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

O `apt-show-versions` mostra que o arquivo `logrotate_3.5.9-8_i386.deb` é a última versão disponível do pacote `logrotate`, então o `logrotate_3.5.9-7_i386.deb` é inútil. O arquivo `gpm_1.19.6-11_i386.deb` também é inútil, já que uma versão mais nova do pacote pode ser baixada.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Finalmente, o `apt-get autoclean` remove apenas os arquivos velhos. Veja 'Como atualizar os pacotes de versões específicas do Debian' on page 16 para obter mais informações sobre o apt-show-versions.

3.7 Usando em conjunto com o Dselect

O `dselect` é um programa que ajuda na seleção de pacotes do Debian. Ele é considerado meio complicado e chato de se lidar, mas com alguma prática, sua interface ncurses, de console, se torna usual.

Uma de suas qualidades é que ele "sabe" como explorar a capacidade de os pacotes "recomendarem" e "sugerirem" a instalação de outros pacotes. Para usá-lo, rode `'dselect'` como root. Selecione como método de acesso o apt. Isso não é realmente necessário mas, caso você não esteja usando um CDROM e quiser usar a internet é o melhor jeito de usar o dselect.

Para entender melhor como usar o dselect, leia a documentação do dselect que se encontra na página do Debian `http://www.debian.org/doc/ddp` ou sua versão em português, encontrada na página do Debian-BR `http://debian-br.cipsga.org.br/suporte/documentacao.html`.

Depois de feitas as seleções adequadas no dselect, use:

```
# apt-get -u dselect-upgrade
```

Veja esse exemplo:

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
```

```
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Veja, no mesmo sistema, se eu rodar apt-get dist-upgrade o que eu tenho:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Note que muitos dos pacotes do de cima estão sendo instalados porque outros pacotes os "sugerem" ou "recomendam". Outros deles estão sendo instalados ou removidos (no caso do lbxproxy por exemplo) por escolha minha, durante minha navegação pela lista de pacotes do dselect. O dselect pode ser uma poderosa ferramenta usada em conjunto com o APT.

3.8 Como manter um sistema misto

As pessoas têm o interesse, as vezes, de usar uma das distribuições Debian como principal e utilizar um ou mais pacotes de uma outra versão.

Para definir qual sua versão principal do Debian, você deve editar o arquivo `/etc/apt/apt.conf` (ele normalmente não existe, crie o arquivo caso não exista) e adicionar a ele:

```
APT::Default-Release "versão";
```

Onde *versão* é a versão do Debian que você quer usar como principal. Os nomes aceitos para 'versão' são: `stable`, `testing`, `unstable`. Para instalar pacotes de uma outra versão, então, você usa o APT da seguinte forma:

```
# apt-get -t distribuição install pacote
```

Para que isso funcione, no ententanto, é necessário haver pelo menos uma fonte do APT listada no arquivo `/etc/apt/sources.list` para a distribuição de que você quer o pacote, e que ela contenha o pacote pedido.

Você pode também pedir a instalação de uma versão específica com a seguinte sintaxe:

```
# apt-get install pacote=versão
```

Por exemplo, a linha abaixo irá instalar a versão `2.2.4-1` do pacote `nautilus`:

```
# apt-get install nautilus=2.2.4-1
```

IMPORTANTE: a versão 'unstable' do Debian é a versão na qual as últimas versões dos pacotes Debian entram. Essa distribuição recebe todas as mudanças pelas quais os pacotes passam, desde as pequenas até as drásticas, que afetam muitos pacotes ou todo o sistema. Por essa razão, essa versão *não* deve ser usada por usuários inexperientes ou que precisem de estabilidade à toda prova.

A versão 'testing' não é necessariamente melhor que a 'unstable', porque não recebe atualizações de segurança rapidamente. Para servidores e outros sistemas de produção deve-se usar a versão estável, sempre.

3.9 Como atualizar os pacotes de versões específicas do Debian

Você usa uma distribuição mixada e agora quer atualizar os pacotes de uma dela. O `apt-show-versions` entra aqui. Você pode usar uma linha como a seguinte para atingir esse objetivo, depois de instalar o pacote `apt-show-versions`:

```
# apt-get install `apt-show-versions -u -b | grep unstable | cut -d ' ' -f 1`
```

3.10 Como manter versões específicas de pacotes instaladas (complexo)

Algumas vezes você fez uma modificação em um programa e não tem tempo ou vontade de portar aquelas mudanças para um versão nova do programa. Então você "prega" a versão que você tem instalada para que não seja feita a atualização. Ou você acaba de atualizar sua distribuição Debian para a 3.0 mas quer continuar com um certo pacote da 2.2.

É para esse propósito que serve o "pinning". A utilização desse recurso é simples. Basta editar o arquivo `/etc/apt/preferences`.

O formato é simples:

```
Package: <pacote>
Pin: <definição do pin>
Pin-Priority: <prioridade do pin>
```

Cada entrada deve ser separada de qualquer outra por uma linha em branco. Por exemplo, para manter o pacote `sylpheed` que eu modifiquei para aceitar "responder para a lista" na versão 0.4.99, eu adiciono:

```
Package: sylpheed
Pin: version 0.4.99*
```

Note que eu usei um * (asterisco). Isso serve para dizer que quero que esse "pin" sirva para todas as versões que comecem com 0.4.99. Isso porque o Debian versiona seus pacotes com uma "revisão Debian" e eu não quero impedir essas revisões de entrar. Ou seja, as versões 0.4.99-1 e 0.4.99-10 seriam instaladas a partir do momento em que existissem. Se você modificou uma versão do pacote você não vai querer que isso aconteça assim, no entanto.

A prioridade do pin ajuda a determinar se um pacote que case com as linhas "Packages:" e "Pin:" será instalado, com as prioridades maiores aumentando a possibilidade de que isso aconteça. Você pode ler `apt_preferences(7)` para uma discussão detalhada de prioridades, mas uns poucos exemplos devem dar uma boa idéia básica. Vejamos uma lista descrevendo o efeito de definir a opção do campo prioridade para diferentes valores no exemplo do sylpheed acima.

1001 A versão 0.4.99 do Sylpheed nunca será trocada pelo apt. Se estiver disponível o APT irá instalar a versão 0.4.99 mesmo que ela substitua um pacote já instalado com uma versão maior. Apenas pacotes com prioridade maior que 1000 terão sua versão rebaixada.

1000 O efeito é o mesmo da prioridade 1001, mas o apt se recusará a baixar uma versão instalada que seja maior que 0.4.99.

990 A versão 0.4.99 será trocada apenas por uma versão maior que esteja disponível a partir da distribuição designada como a preferida usando a variável "APT::Default-Release" (veja 'Como manter um sistema misto' on page 15 acima).

500 Qualquer versão maior que 0.4.99 do sylpheed que esteja disponível a partir de qualquer distribuição terá preferência sobre a 0.4.99, mas a 0.4.99 será ainda preferida a qualquer versão inferior.

100 Versões maiores do sylpheed disponíveis de qualquer distribuição terão preferência sobre a versão 0.4.99, assim como qualquer versão maior que esteja instalada; então a versão 0.4.99 será instalada somente quando não houver versão alguma instalada. Essa é a prioridade dos pacotes instalados.

-1 As prioridades negativas são permitidas, também, e evitam que a versão 0.4.99 seja instalada.

As opções para o pin podem ser: `version`, `release` ou `origin`.

A opção `version`, como já vimos, suporta uma versão normal e uma máscara (como um asterisco, por exemplo) para definir várias verões de uma vez.

A opção `release` é mais ampla e depende do arquivo Release do repositório APT, ou do CD. Esta opção pode deixar a desejar pois alguns repositórios não o contém. Você pode ver o conteúdo dos arquivos Release que você tem em `/var/lib/apt/lists/`. As sub-opções são: a (archive), c (componente), v (versão), o (origin) e l (label).

Um exemplo:

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

Neste exemplo escolhemos versão do Debian 2.2*, que considera as "revisões" (que vêm com consertos de segurança e bugs sérios), repositório `stable`, seção `main` (poderia ser contrib e non-free, por exemplo) e origem e nome Debian. Origem (o=) define quem produziu aquele arquivo Release, o nome (l=) define o nome da distribuição: Debian para o próprio Debian e Progeny para a mesma, por exemplo. Um exemplo de arquivo Release:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i3
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Capítulo 4

Ajudantes muito úteis

4.1 Como instalar programas compilados localmente: equivs

As vezes as pessoas querem usar uma versão específica para um programa, uma versão mais nova, só disponível em fontes, sem pacote Debian. Mas o sistema de empacotamento pode atrapalhar esses planos. Suponha que você quer compilar uma nova versão do seu servidor de email. Tudo vai bem, mas inúmeros pacotes no Debian dependem de um MTA (Mail Transport Agent) instalado. Como você instalou algo que você mesmo compilou, o sistema de pacotes não sabe da presença dele.

É aí que entra o `equivs`. Para usá-lo instale o pacote de mesmo nome. O equivs cria um pacote vazio que preenche dependências, fazendo com que o sistema de pacotes acredite que as dependências estão satisfeitas.

Antes de continuarmos, é bom lembrar que há maneiras mais seguras de se compilar um programa que já está debianizado com opções alteradas, e que não se deve usar o equivs para substituir dependências se você não souber realmente o que está fazendo. Veja a seção 'Lidando com pacotes fonte' on page 29 para saber mais.

Vamos continuar com o exemplo do MTA, você acaba de instalar seu novo `postfix` compilado e parte para a instalação do `mutt`. Qual não é sua surpresa quando descobre que o `mutt` quer instalar um outro MTA, mas você já tem o seu.

Vá para um diretório qualquer (`/tmp`, por exemplo) e execute:

```
# equivs-control nome
```

Substitua *nome* pelo nome do arquivo de controle que você quer criar. O arquivo vai ser criado da seguinte forma:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1
```

```
Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, comma separated>
Description: <short description; defaults to some wise words>
 long description and info
 .
 second paragraph
```

Basta alterar isso agora para que o que queremos seja feito. Observe bem o formato dos campos e as descrições deles, não há necessidade aqui de discorrer sobre cada um desses, vamos ao que é necessário:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

Sim, é só isso. O `mutt` depende de `mail-transport-agent`, que é um pacote virtual fornecido por todos os MTAs, eu poderia simplesmente chamar o pacote de `mail-transport-agent`, também, mas preferi usar o esquema do pacote virtual,usando o Provides.

Agora basta construir o pacote:

```
# equivs-build nome
dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
```

```
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package 'nome' in '../nome_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

E instalar o `.deb` resultante.

Como se pode ver, há inúmeras utilidades para o `equivs`. Inclusive criar um pacote `meus-preferidos`, que dependa dos programas que você sempre instala, por exemplo. É soltar a imaginação, mas com cuidado.

É importante notar que há exemplos de arquivos de controle em `/usr/share/doc/equivs/examples`. Confira.

4.2 Removendo arquivos de locale não usados: localepurge

Muitos usuários de Debian usam apenas um locale. Um usuário de Debian brasileiro, por exemplo, normalmente usa o locale `pt_BR` o tempo todo e não se importa com o `es`.

O `localepurge` é uma ferramenta muito útil para esses usuários. Você pode liberar muito espaço tendo apenas os locales que você realmente usa. Basta usar `apt-get install localepurge`.

É muito fácil configurá-lo, perguntas feitas com o debconf guiam o usuário em uma configuração passo-a-passo. Seja cuidadoso ao responder a primeira questão; respostas erradas podem remover todos os arquivos de locale, mesmo aqueles que você usa. A única maneira de recuperá-los será reinstalar todos os pacotes que os fornecem.

4.3 Como saber quais pacotes podem ser atualizados

O `apt-show-versions` é um programa que mostra quais pacotes do seu sistema podem ser atualizados e várias outras informações úteis. Para saber quais pacotes podem ser atualizados faça:

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Capítulo 5

Obtendo informações sobre os pacotes.

Existem alguns programas, conhecidos como `front-ends` para o sistema APT que facilitam em muito a listagem dos pacotes disponíveis para instalação, os instalados, em quais seções se encontram, quais suas prioridades, descrições, etc.

Mas... nossa intenção aqui é aprender a usar o APT puro. Então o que fazer para descobrir o nome do pacote que você pode querer a vir instalar?

Para isso temos uma série de recursos. Vamos começar com o `apt-cache`. Esse programa é usado pelo sistema APT para manter seu banco de dados. Nós vamos entrar apenas nos aspectos práticos dele.

5.1 Descobrindo nome dos pacotes

Por exemplo, suponha que você está com vontade de relembrar os bons tempos do atari 2600. Você quer usar o APT para instalar um emulador de atari e depois baixar alguns jogos. Você pode fazer:

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Achamos então vários pacotes relacionados ao que queremos e breves descrições. Para ter mais informações sobre um pacote, posso, então, usar:

```
# apt-cache show stella
```

```
Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Nessa saída você tem inúmeras informações sobre o pacote que quer instalar (ou não) e a sua descrição completa. No caso de o pacote já estar instalado em seu sistema e haver uma versão mais nova, você verá informações sobre ambos. Por exemplo:

```
# apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
```

```
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Note, que a primeira a ser listada é a disponível e a segunda, a que já se encontra instalada.
Para uma informação mais geral sobre o pacote, você usa:

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_mai

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libs
Provides:
1.4.5-1 -
Reverse Provides:
```

Para saber de quais pacotes ele depende, apenas:

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

Resumindo, temos um bom arsenal para encontrarmos o nome do pacote que queremos.

5.2 Usando o dpkg para achar nomes de pacotes

Uma das maneiras de localizar o nome de um pacote é saber o nome de um arquivo importante que está nesse pacote. Por exemplo, para achar o pacote ao qual pertence um arquivo "`.h`"

necessário a uma compilação você pode fazer:

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h

ou:

# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Para saber o nome de pacotes instalados no seu sistema, para uma possível limpeza, por exemplo, você pode usar:

```
# dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

O problema com esse comando é que ele pode "quebrar" o nome do pacote. No exemplo acima o nome todo do pacote é `mozilla-browser`. Para dar um jeito nisso, você pode usar a variável de ambiente `COLUMNS`, assim, por exemplo:

```
$ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                      Mozilla Web Brows
```

ou usar a descrição assim:

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 Como instalar pacotes "on demand"

Você está compilando um programa e, de repente, boom! Há um erro porque falta um arquivo `.h` que você não tem. O `auto-apt` pode te salvar de coisas assim. Ele pede para instalar os pacotes caso sejam necessários pausando o processo dependente e depois de instalado o pacote, continuando.

O que você faz, basicamente, é executar:

```
# auto-apt run comando
```

Onde 'comando' é o comando a ser executado que pode vir a precisar de um arquivo qualquer. Por exemplo:

```
# auto-apt run ./configure
```

Ele irá então pedir para instalar os pacotes necessários e irá chamar o apt-get automaticamente. Caso você esteja no X, uma interface gráfica substituirá a interface de texto padrão.

O auto-apt mantém bancos de dados que devem ser atualizados para que ele tenha real eficácia, isso é feito chamando os comandos `auto-apt update`, `auto-apt updatedb` e `auto-apt update-local`.

5.4 Como descobrir a qual pacote um arquivo pertence

Se você quer instalar um pacote e não consegue achar o seu nome usando o `apt-cache` mas sabe o nome do arquivo do programa ou de algum outro arquivo que pertence ao pacote pode usar o `apt-file` para encontrar o nome do pacote. Isso é feito assim:

```
$ apt-file search nomedoarquivo
```

Ele funciona do mesmo jeito que o `dpkg -S`, mas também mostra os pacotes não instalados que contêm o arquivo. Ele poderia também ser usado para encontrar que pacotes contêm arquivos include necessários para uma compilação, apesar de o `auto-apt` ser um método muito melhor para resolver tais problemas, veja 'Como instalar pacotes "on demand"' on the facing page.

Você pode também listar o conteúdo de um pacote rodando:

```
$ apt-file list nomedopacote
```

O `apt-file` mantém uma base de dados de quais arquivos cada pacote contém, assim como o auto-apt e essa base precisa estar atualizada. Isso é feito rodando:

```
# apt-file update
```

Por padrão, o `apt-file` usa a mesma base de dados que o `auto-apt` está usando, veja 'Como instalar pacotes "on demand"' on the preceding page.

5.5 Como manter-se informado das mudanças nos pacotes.

Todo pacote instala em seu diretório de documentação (`/usr/share/doc/nomedopacote`) um arquivo chamado `changelog.Debian.gz` que contém a lista de mudanças feitas no pacote desde a última versão. Você pode ler esses arquivos com a ajuda do `zless`, por exemplo, mas é algo pouco prático acabar de instalar todos os pacotes de uma atualização completa do sistema e então sair procurando os changelogs deles todos.

Há um jeito de automatizar essa tarefa por meio da ferramenta chamada `apt-listchanges`. Para começar instala-se o pacote `apt-listchanges`. Durante a instalação do pacote uma configuração será feita com o Debconf. Algumas perguntas podem não ser exibidas dependendo da prioridade que você configurou para seu Debconf usar. Responda às perguntas de acordo com sua vontade.

A primeira pergunta é sobre a forma com que você quer que as mudanças sejam exibidas. Você pode fazer com que sejam enviadas por e-mail para você, que é bom para atualizações automatizadas ou você pode pedir que sejam mostradas em um paginador como o `less`, para que você possa inspecionar as mudanças antes de permitir que a atualização continue. Se você não quiser que o `apt-listchanges` rode automaticamente durante atualizações você pode responder `none`.

Depois do apt-listchanges instalado, logo depois de ter baixado todos os arquivos (ou tê-los adquirido de um CD ou disco montado) o apt irá mostrar as listas de mudanças ocorridas naqueles pacotes antes de iniciar a instalação.

Capítulo 6

Lidando com pacotes fonte

6.1 Baixando pacotes fonte

É comum no mundo livre que se estude código fonte ou mesmo que se faça correções em código fonte com erros. Para isso é necessário que se baixe o fonte do programa. O sistema APT provê uma maneira fácil de se obter os fontes dos vários programas contidos na distribuição, com, inclusive, os arquivos necessários para se criar o .deb do programa.

Outro uso comum dos fontes no Debian é o de adequar uma versão mais nova de um programa que está na distribuição unstable, por exemplo, à estável. Compilar um pacote no stable gera .debs com dependências ajustadas aos pacotes disponíveis nessa distribuição.

Para que isso seja feito, a entrada `deb-src` do seu `/etc/apt/sources.list` deve estar apontando para a unstable. E esteja habilitada (descomentada). Veja a seção 'O arquivo /etc/apt/sources.list' on page 3.

Para baixar um pacote fonte, você usa o seguinte comando:

```
$ apt-get source nomedopacote
```

Ele irá baixar três arquivos. Um `.orig.tar.gz`, um `.dsc` e um `.diff.gz`. No caso de pacotes feitos especificamente para o Debian, o último desses não é baixado e o primeiro costuma não ter a parte "`orig`" no nome.

O arquivo `.dsc` é usado pelo dpkg-source para descompactar o pacote fonte no diretório *nomedopacote-versão*. Dentro de todo pacote fonte baixado existe o diretorio `debian/` que contém os arquivos para se criar o pacote .deb.

Para que o pacote seja auto-construído ao acabar de ser baixado, basta especificar `-b` na linha de comando, assim:

```
$ apt-get -b source nomedopacote
```

Se você decidir não gerar o .deb logo após o download, você pode fazê-lo depois rodando:

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

de dentro do diretório criado para o pacote após o download. Para instalar um arquivo `.deb` deve-se usar o gerenciador de pacotes diretamente, da seguinte forma:

```
# dpkg -i arquivo.deb
```

Há uma grande diferença entre o método `source` do `apt-get` e os outros. Este pode ser usado por usuários comuns, sem a necessidade de poderes especiais de root. Os arquivos são baixados no diretório de onde foi chamado o comando `apt-get source pacote`.

6.2 Pacotes necessários para compilação de um pacote fonte

Pacotes fontes, normalmente precisam de bibliotecas compartilhadas e headers específicos para serem compilados. Todo pacote fonte tem em seus arquivos de controle um campo conhecido como 'Build-Depends:' que indica quais os pacotes são necessários para se compilar o pacote fonte.

O APT tem uma maneira simples de baixar esses pacotes, basta você executar `apt-get build-dep pacote`, onde 'pacote' é o nome do pacote que você vai construir. Por exemplo:

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-de
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Os pacotes que serão instalados são os pacotes dos quais o programa `gmc` necessita para se construir perfeitamente. É importante notar que esse comando não busca o pacote fonte do programa a ser compilado. Assim, você deve rodar o `apt-get source` separadamente para obtê-lo.

Para apenas ver quais pacotes são necessários para a compilação de determinado pacote existe uma variante do comando `apt-cache show` (veja 'Obtendo informações sobre os pacotes.' on page 23), que irá mostrar, entre outras informações, a linha `Build-Depends` que lista esses pacotes:

```
# apt-cache showsrc pacote
```

Capítulo 7

Como lidar com erros?

7.1 Erros comuns

Erros sempre acontecem, muitos deles causados por falta de atenção do usuário. Aqui estou listando alguns dos erros reportados mais frequentemente e como lidar com eles.

Caso você receba uma mensagem parecida com a seguinte ao tentar rodar o `apt-get install pacote`...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

você se esqueceu de rodar o `apt-get update` depois da sua última alteração no arquivo `/etc/apt/sources.list`.

Se o erro for parecido com:

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

ao tentar qualquer dos métods do `apt-get` com excessão do `source`, você não tem permissão root, ou seja, está acessando como usuário normal.

Há um erro parecido com o de cima que acontece ao rodar duas instâncias do `apt-get` juntas ou mesmo ao tentar rodar o `apt-get` enquanto há um processo do `dpkg` ativo. O único método que pode rodar simultâneamente com outro é o `source`.

Caso uma instalação quebre durante o processo e você fique impossibilitado de instalar e remover pacotes, tente rodar esses dois comandos:

```
# apt-get -f install
# dpkg --configure -a
```

E tente o processo novamente, pode ser preciso rodar o segundo comando acima mais vezes. Essa é uma dica importante aos aventureiros que usam 'unstable'.

Se você receber o erro "E: Dynamic MMap ran out of room" ao rodar `apt-get update`, adicione a linha seguinte ao `/etc/apt/apt.conf`:

```
APT::Cache-Limit 10000000;
```

7.2 Onde consigo ajuda?

Caso alguma dúvida cruel venha à sua cabeça, consulte a extensiva documentação existente sobre o sistema de empacotamento do Debian. `--help`'s e manpages podem ser de enorme ajuda para você, assim como a documentação contida nos diretórios do `/usr/share/doc` como o `apt`.

Se mesmo assim a dúvida persistir, consulte as listas do Debian por uma resposta. Você pode conseguir mais informações sobre a lista específica para usuários falantes da língua portuguesa em: http://debian-br.cipsga.org.br, o site do projeto Debian-BR. Outras listas do Debian podem ser encontradas na página do Debian em: http://www.debian.org. Uma boa parte dessa página se encontra traduzida graças ao esforço do Debian-BR.

Lembre-se de que essas listas e recursos devem ser usados apenas por usuários do Debian, usuários de outros sistemas devem encontrar um melhor suporte junto à suas distribuições.

Capítulo 8

Quais são as distribuições que suportam o APT?

Aqui estão os nomes de algumas das distribuições que contam com o APT:

Debian GNU/Linux (http://www.debian.org) - foi para esta distribuição que o APT foi desenvolvido

Conectiva (http://www.conectiva.com.br) - essa foi a primeira distribuição a portar esse software para lidar com rpm

Libranet (http://www.libranet.com)

Mandrake (http://www.mandrake.com)

PLD (http://www.pld.org.pl)

Vine (http://www.vinelinux.org)

APT4RPM (http://apt4rpm.sf.net)

Alt Linux (http://www.altlinux.ru/)

Red Hat (http://www.redhat.com/)

Sun Solaris (http://www.sun.com/)

SuSE (http://www.suse.de/)

Yellow Dog Linux (http://www.yellowdoglinux.com/)

Capítulo 9

Agradecimentos

Agradecimentos vão para meus grandes amigos do projeto Debian-BR e do Debian que me ajudam sempre e me dão força para continuar trabalhando em prol da humanidade além de me ajudar sempre no meu objetivo de salvar o mundo =).

Agradeço também ao CIPSGA pelo apoio enorme que dá ao nosso projeto e a todos os projetos livres que brotam de grandes idéias.

E agradecimentos especiais para:

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <pawel.tecza@poczta.fm>

Hugo Mora <h.mora@melix.com.mx>

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzzoni <spectra@debian.org>

Steve Langasek <vorlon@netexpress.net>

Arnaldo Carvalho de Melo <acme@conectiva.com.br>

Erik Rossen <rossen@freesurf.ch>

Ross Boylan <RossBoylan@stanfordalumni.org>

Matt Kraai <kraai@debian.org>

Aaron M. Ucko <ucko@debian.org>

Jon Åslund <d98-jas@nada.kth.se>

Capítulo 10

Novas versões desse tutorial

Esse manual foi produzido pelo projeto Debian-BR (http://www.debian-br.org), com o intuito de ajudar no uso cotidiano do sistema Debian.

Novas versões serão lançadas na página do Projeto, em http://www.debian-br.org/suporte/documentacao.php.

Sugestões e críticas podem ser enviadas diretamente para mim no email <kov@debian.org>.

Um abraço a toda a comunidade Debian!