5a. Parte:

Gerenciamento de Programas

Índice

ABERTURA

Diferente do gerenciamento de programas do *Windows*, nos sistemas *GNU/Linux* – e em especial no *Slackware* – existe uma certa necessidade de obtermos conhecimentos técnicos e dominar certas operações relacionadas, onde entra em destaque a famosa questão das pendências.

Os aplicativos existentes para os sistemas *GNU/Linux* geralmente são disponibilizados em um único arquivo chamado pacote. Como o próprio nome diz, um pacote é o conjunto de arquivos componentes de um programa, arquivado em um formato especial reconhecido pelo gerenciador de pacotes da distribuição.

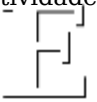
O gerenciamento de programas em qualquer sistema operacional, por mais simples que seja, geralmente causam diversas dúvidas tais como conflitos, formas de instalação, atualizações, bibliotecas, etc. Em sistemas *GNU/Linux*, é lógico que não poderia ser diferente e, ao invés dos pacotes serem disponibilizados com todos os requerimentos adicionais, estes são encontrados contendo somente o programa principal, necessitando da instalação de outros pacotes adicionais para que possam serem executados corretamente. Estes pacotes indispensáveis são chamados de pendências.

As pendências possuem diversas características e finalidades, porém a grande maioria possuem em comum a necessidade de sua presença para o correto funcionamento do aplicativo. Existem diversas categorias de pendências necessárias, mas as principais são as bibliotecas do sistema. No *Windows*, temos as famosas *DLL*; já nos sistemas *GNU/Linux*, estas são substituídas por bibliotecas específicas do sistema, que possuem os mesmos conceitos e finalidades (além de outras mais).

Cada tipo de aplicação requer uma ou um conjunto específico de bibliotecas. Programas multimídia geralmente necessitam de bibliotecas para áudio e vídeo; programas de tratamento de imagens já requerem bibliotecas específicas para a leitura de diversos formatos; os jogos utilizam largamente bibliotecas de acesso à periféricos como a placa de som e vídeo – famosamente conhecida como *APIs*.

Geralmente todas as instruções necessárias para obter informações sobre as pendências necessárias encontram-se na página oficial do programa e/ou na documentação contida em seu próprio pacote (*README* e *INSTALL*), onde uma consulta básica pode resolver a maioria dos problemas e anomalias que venham porventura ocorrer.

Nos próximos capítulos conheceremos a fundo todo o processo de gerenciamentos de programas no *Slackware*, passando por diversos aspectos, desde a utilização de ferramentas nativas a outros métodos e técnicas de instalação, atualização e remoção, com ênfase também na utilização do gerenciador de pacotes *RPM*, na conversão de pacotes, atualização, compilação e outras atividades necessárias. &;-D



I. Ferramentas de gerenciamento

Introdução

Não muito diferente das demais distribuições, o *Slackware* possui também seu próprio gerenciador de pacotes pré-compilados, além de diversas ferramentas – apesar da inexistência de funcionalidades importantes como a checagem de dependências. Neste capítulo iremos conhecê-las, onde entra em destaque, o *Slackware Package Tools* – conhecido popularmente como *Pkgtool*, um utilitário em modo texto simples e fácil de utilizar.

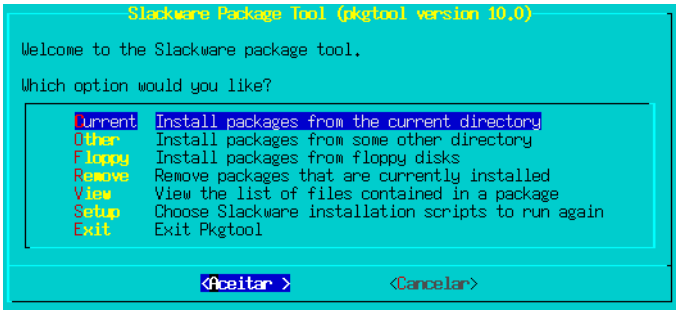
As ferramentas

Slackware Package Tools

O *Slackware Package Tools* – o chamaremos aqui como *Pkgtool* – é o gerenciador de pacotes padrão do *Slackware*, escrito basicamente com a utilização de *scripts* (arquivos de lote). Além das funcionalidades básicas necessárias para o gerenciamento de pacotes, o *Pkgtool* também provê uma série de *scripts* de configuração que auxiliam muito para a realização de manutenções gerais do sistema.

Para executar o utilitário, deveremos carregá-lo na linha de comando...

```
# pkgtool
```



Interface principal do Pkgtool.

O *Pkgtool* provê ao administrador uma interface texto com menus intuitivos e fácil de utilizar. Basta apenas utilizar as opções básicas que se encontram em sua interface, das quais seguem:

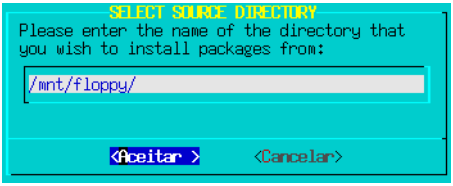
Current

Instala os pacotes pertencentes ao diretório onde este utilitário é invocado. Por exemplo, se entrarmos em nosso diretório */mnt/pkg* e utilizarmos esta opção, todos os pacotes nativos do *Slackware* presentes neste diretório

serão instalados automaticamente. É muito útil para a instalação de pacotes extras de forma simples e automatizada, bastando apenas guardá-los em um diretório separado de acordo com o perfil utilizado.

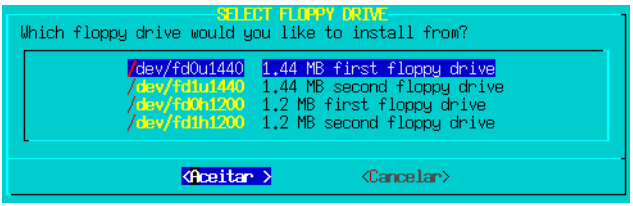
Other

Possui as mesma finalidade da opção *Current*, porém solicita ao superusuário o endereço do diretório o qual contém os pacotes desejados para a instalação.



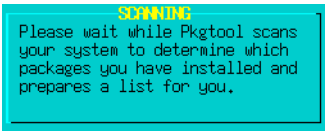
Floppy

Realiza a instalação dos pacotes desejados, porém estes tendo como origem a unidade de disquetes. Basta selecionar a unidade que possui o disquete com os pacotes desejados.

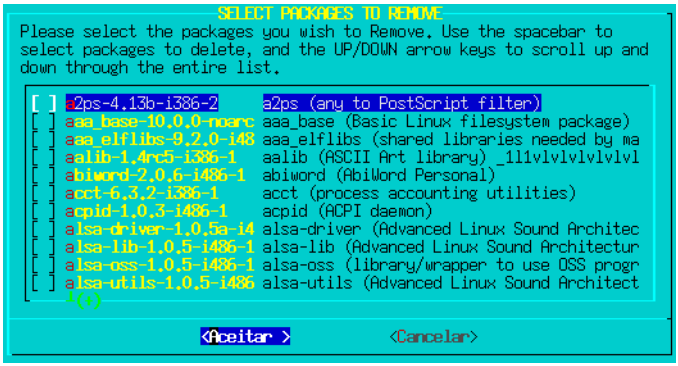


Remove

Remove os pacotes desejados conforme uma seleção pré-realizada.



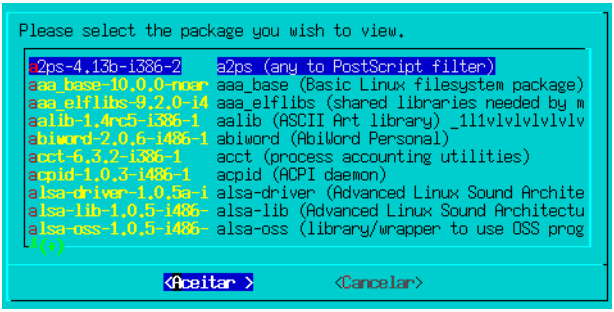
> *"Por favor espere enquanto Pkgtool procura em seu sistema para determinar quais pacotes você têm instalado e preparar uma lista para você." -> [Tradução]*



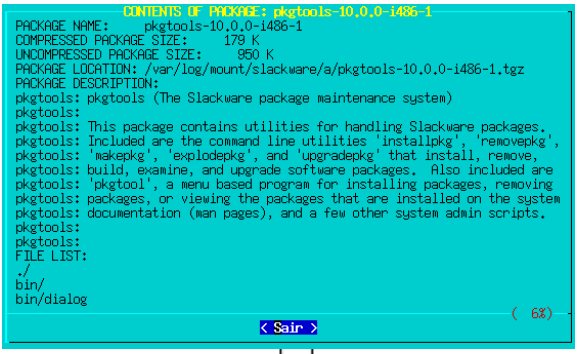
Acionando a <BARRA_DE_ESPAÇO>, marcaremos e/ou desmarcaremos os pacotes que desejamos desinstalar e/ou manter.

View

Exibe um conjunto de informações referentes ao pacote selecionado, onde será mostrado uma listagem de pacotes instalados no sistema.



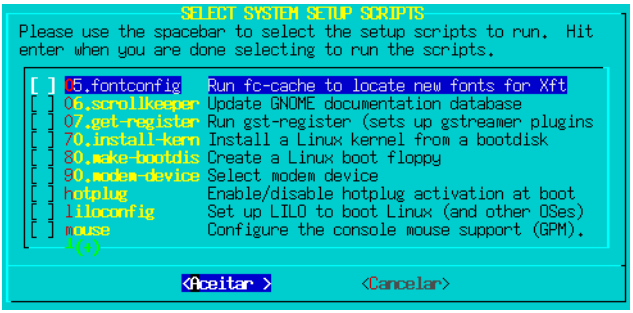
Porém, poderá ser realizado o acesso das informações somente por um pacote de cada vez. Teclando <ENTER>...



... obteremos as informações desejadas, e com o uso das teclas <SETA ACIMA> e <SETA_ABAIXO> faremos a rolagem do texto.

Setup

Da mesma forma que na instalação do *Slackware*, a opção *setup* disponibiliza uma série de *scripts* para a configuração.



Deveremos marcá-los com a tecla <BARRA_DE_ESPAÇO> e teclar <ENTER> em seguida para que eles sejam executadas seqüencialmente. Aqui encontram-se as seguintes opções:

Setup	
Fontconfig	Instalação e configuração das fontes do sistema.
Install-kernel	Instalação de diferentes versões compiladas do *kernel*.
Make-bootdisk	Criação de discos de inicialização para o sistema.
Moden-device	Ajustes de device e permissões de *hardmodens*.
Hotplug	Auto-detecção de periféricos.
Liloconfig	Ajustes e configurações do *LILO*.
Mouse	Configuração do *mouse*.
Netconfig	Configuração da rede.
Services	Habilita / desabilita os serviços disponíveis.
Setconsolefont	Ajustes das fontes do console.
Timeconfig	Ajuste do fuso horário.
Xwmnconfig	Seleção do ambiente gráfico padrão do sistema.

Poderemos obter mais informações adicionais sobre a utilização destas sub-opções durante a consulta dos capítulos referente à configuração geral do sistema, entre outros. Estas informações situam-se na *4a. Parte: Ajustes & Configurações*.

Ferramentas de linha de comando

Além do *Slackware Package Tools*, temos também várias outras ferramentas em linha de comando, dos quais compreende os seguintes comandos: *installpkg*, *removepkg*, *upgradepkg*, *explodepkg* e *makepkg*.

Instalação

Para instalar pacotes, utilizamos o comando *installpkg*. Como o próprio nome diz, este utilitário é utilizado para a instalação de pacotes pré-compilados.

Sintaxe:

```
# installpkg [PACOTE]-[VERSÃO]-[PLATAFORMA].tgz
```

Exemplo:

```
# installpkg libdvdread-0.9.4-i686-1.tgz

Installing package libdvdread-0.9.4-i686-1...
PACKAGE DESCRIPTION:
libdvdread: libdvdread (DVD access library)
libdvdread:
libdvdread: libdvdread provides a simple foundation for reading DVD video disks.
libdvdread: It provides the functionality that is required to access many DVDs.
libdvdread: It parses IFO files, reads NAV-blocks, and performs CSS
libdvdread: authentication and descrambling.
Libdvdread:
Executing install script for libdvdread-0.9.4-i686-1...

# _
```

Como podem ver, durante a instalação do pacote, é mostrada um conjunto de informações do pacote instalado. Atentem-se para a nomenclatura do arquivo, o qual indica a versão do programa (*0.9.4*) e plataforma onde foi compilado (*i686*), neste caso otimizado para *Pentium II*.

Remoção

Da mesma forma que o comando *installpkg*, utilizamos o comando *removepkg*, que tem a função de remover o pacote desejado.

Sintaxe:

```
# removepkg [PACOTE]
```

Exemplo:

Observem que a definição da versão, plataforma e extensão na sintaxe deste comando é desnecessária, bastando apenas saber o nome do pacote que se deseja desinstalar.

```
# removepkg libdvdread

Removing package /var/log/packages/libdvdread-0.9.4-i686-1...
Removing files:
  --> Deleting symlink /usr/lib/libdvdread.so
```

```
  --> Deleting symlink /usr/lib/libdvdread.so.3
  --> Deleting /usr/doc/libdvdread-0.9.4/AUTHORS
  --> Deleting /usr/doc/libdvdread-0.9.4/ChangeLog
  --> Deleting /usr/doc/libdvdread-0.9.4/COPYING
  --> Deleting /usr/doc/libdvdread-0.9.4/INSTALL
  --> Deleting /usr/doc/libdvdread-0.9.4/NEWS
  --> Deleting /usr/doc/libdvdread-0.9.4/README
  --> Deleting /usr/doc/libdvdread-0.9.4/TODO
  --> Deleting /usr/include/dvdread/dvd_reader.h
  --> Deleting /usr/include/dvdread/ifo_print.h
  --> Deleting /usr/include/dvdread/ifo_read.h
  --> Deleting /usr/include/dvdread/ifo_types.h
  --> Deleting /usr/include/dvdread/nav_print.h
  --> Deleting /usr/include/dvdread/nav_read.h
  --> Deleting /usr/include/dvdread/nav_types.h
  --> Deleting /usr/lib/libdvdread.a
  --> Deleting /usr/lib/libdvdread.la
  --> Deleting /usr/lib/libdvdread.so.3.0.0
  --> Deleting empty directory /usr/include/dvdread/
  --> Deleting empty directory /usr/doc/libdvdread-0.9.4/
# _
```

Simples, não? Observe ainda que é mantido em */var/log/packages/* um arquivo texto com o nome do pacote contendo todas as informações gerais.

Atualização

Para esta função, utilizamos o comando *upgradepkg*.

Sintaxe:

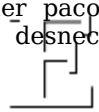
```
# upgradepkg [PACOTE]-[VERSÃO]-[PLATAFORMA].tgz
```

Este comando apenas atualiza um único pacote ou conjunto dele (desde que situados em um mesmo diretório). Mas não é o ideal utilizá-lo para atualizar todos os pacotes do sistema. Para esta atividade, temos ótimas ferramentas.

Recomendações gerais

Uma das características marcantes do *Slackware* está no fato de que seu gerenciador de pacotes padrão não possuir o famoso recurso de detecção de pendência. Para os iniciantes, isto pode se tornar uma grande dor de cabeça em virtude de suas poucas experiências em lidar com a instalação de programas. Face à isto, temos algumas simples recomendações à fazer:

1. Prefiram sempre realizar a instalação completa (*FULL*);
2. Consultem na página oficial, no arquivo *README* presente no código-fonte, ou na documentação do pacote em */usr/doc/[PACOTE]/*, quais são as pendências necessárias – se houverem, instalem-nas primeiro;
3. Evitem remover quaisquer pacotes (especialmente bibliotecas e *APIs*) que considerarem desnecessários, ao menos que saibam



EXATAMENTE o que estão fazendo. Além disso, alguns programas também são necessários para a execução de outros aplicativos.

Apesar da existência de utilitários externos que possibilitem a verificação de pendência de pacotes (como o *Swaret*), a maioria destes ainda se encontram em um estágio imaturo ou de poucos e limitados recursos. Como isto, em muitas circunstâncias estes poderão ser ineficientes, especialmente com pacotes não pertencentes à distribuição.

Já a compilação manual realiza a detecção de pendências, onde deveremos ficar atento às mensagens exibidas durante a execução dos processo. Serão exibidos na saída de vídeo as pendências gerais e uma notificação *yes/no*, caso se encontrem ou não no sistema. Em caso de pacotes necessários, o processo será abortado até que a pendência esteja satisfeita. Estas mensagens geralmente aparecem na execução do *script ./configure*.

Por último, para consultarmos os pacotes removidos do sistema, verifiquem no diretório */var/log/*. Lá iremos encontrar os diretórios *removed_packages* e *removed_scripts*, onde se encontrarão os arquivos-textos com as informações desejadas sobre estes pacotes.

```
$ ls /var/log/
Xorg.0.log      debug    kdm.log   packages          scrollkeeper.log  uucp
Xorg.0.log.old  dmesg    lastlog   removed_packages  secure            wtmp
apache          faillog  maillog   removed_scripts   setup
cron            gdm      messages  samba             spooler
cups            iptraf   nfsd      scripts           syslog
$ _
```

Conclusão

A administração de programas nunca foi uma tarefa simples, mesmo que muitas dicas e tutoriais venham a salientar estes conceitos. Por mais simples que sejam os comandos e respectivos parâmetros, poderão correr alguns inconvenientes que possam criar dificuldades para a realização de todo o processo. Por isto, sempre tenham em mente a obtenção de informações e conceitos básicos inerentes à desenvoltura destes processos, pois nestas ocasiões, serão eles a base principal para a solução da maioria das dificuldades que possam vir à encontrar. &;-D

II. Ferramentas de atualização

Introdução

Antigamente uma das maiores desvantagens do sistema de gerenciamento de pacotes do *Slackware* estava no fato de não existir nenhuma ferramenta de automação para a atualização de pacotes. Mas felizmente, graças à colaboração da comunidade, hoje tempos ótimas ferramentas desenvolvidas por ela para esta atividade. À partir da versão *9.1*, o *Slackware* foi agraciado com as ferramentas *Slapt-get*, *Slackpkg* e *Swaret*. Nesta literatura, concentraremos nossa atenção no *Slackpkg*.

O Slackpkg

- ✔ <*http://slackpkg.sourceforge.net/*>.

O *Slackpkg*, concebido pelo brasileiro *Roberto F. Batista* (*Piter Punk*) e *Evaldo Gardenali* (*UdontKnow*), é um *script* desenvolvido para a instalação e atualização de pacotes específicos do *Slackware*. Basicamente as atividades deste programa consiste em checar o sistema, verificar quais os pacotes se encontram instalados e baixar suas respectivas atualizações.

A instalação

Para obtermos o *Slackpkg*, deveremos baixá-lo da página oficial do projeto. Podemos também encontrá-lo no *2o. CD-ROM* de instalação distribuição. A *1a.* opção será a mais recomendada por estar atualizada, especialmente os *CD-ROMs* estiverem com alguns meses de idade... &;-D

Por se encontrar disponível o pacote pré-compilado da ferramenta, não há necessidade de fornecer instruções adicionais para a instalação.

A configuração

Após a instalação da ferramenta, será necessário a edição do arquivo de configuração */etc/slackpkg/mirrors*, onde deverá ser definido qual o espelho que será utilizado para obter as atualizações necessárias. Para isto, deveremos desmarcar a listagem presente ou obter uma lista atualizada na página <*http://www.slackware.com/getslack/*>.

```
# mirrors - List of Slackware Linux mirrors.
#
# SlackPkg - An Automated packaging tool for Slackware Linux
# Copyright (C) 2003 Roberto F. Batista, Evaldo Gardenali
-//-
#
# Project Page: http://slackpkg.sf.net/
# Roberto F. Batista (aka PiterPunk) piterpk@terra.com.br
# Evaldo Gardenali (aka UdontKnow) evaldogardenali@fasternet.com.br
```

```
#
# END OF LEGAL NOTICE
# You only need to select one mirror and uncomment them. Please,
-//-
# ONLY ONE mirror can be uncommented each time.

#
# Local CD drive
#
#cdrom://mnt/cdrom/

#
# Slackware [VERSÃO] mirrors, ordered by country
#
-//-
```

O *Slackpkg* utiliza somente um único espelho para obter os pacotes atualizados, apesar de suportar os servidores *HTTP* e *FTP*. Caso contrário, o programa simplesmente não funcionará.

Outro ajuste interessante está no arquivo de configuração *blacklist*:

```
# This is a blacklist file. Any packages listed here won't be
# upgraded, removed or installed by slackpkg.
#
# The correct syntax is:
#
# to blacklist the package xfree86-devel-4.3.0-i386-1 the line will be:
# xfree86-devel
#
# Please, DON'T put any blank line(s) or any space(s) before or
# after the package name.
# If you do this, the blacklist will NOT work.
#
# Automated upgrade of kernel packages aren't a good idea (and you need to
# run "lilo" after upgrade). If you think the same, uncomment the lines
# below
#
#kernel-ide
#kernel-modules
#kernel-source
#kernel-headers
aaa_elflibs
```

Como podem ver, recomenda-se a não atualização destes pacotes para evitar conflitos no sistema. Quanto aos pacotes referentes ao *kernel*, apesar da possibilidade de atualizá-lo, os criados do projeto não o aconselham. Mas se ainda assim desejarem fazê-lo, descomentem as linhas referentes ao *kernel* antes de utilizar a ferramenta. Após isto, executem novamente o *LILO* para que sejam redefinidas as novas configurações. Caso contrário, haverão travamentos que impedirão a inicialização do sistema, nos obrigando à inicializar a máquina com os *CD-ROMs* de instalação para realizar a regravação do *LILO* na *MBR*.

A UTILIZAÇÃO

A sintaxe básica do programa é:

```
# slackpkg [PARÂMETROS] [PACOTE]
```

Onde:

Slackpkg	
install	Para instalar um pacote. `# slackpkg install [PACOTE]`
reinstall	Para reinstalar um pacote. `# slackpkg reinstall [PACOTE]`

upgrade	Para atualizar um pacote.
	`# slackpkg upgrade [PACOTE]`
	Para aplicar uma correção (*patche*).
	`# slackpkg upgrade [CORREÇÃO]`

Remove	Para remover um pacote. `# slackpkg remove [PACOTE]`

Para atualizar todo o sistema, deveremos digitar...

```
# slackpkg update
```

... para atualizar a lista de pacotes.

```
-//-
100%[====================================>] 18,743         5.95K/s    ETA
00:00

22:20:43 (5.94 KB/s) - `/tmp/pasture-PACKAGES.TXT' recebido [18743]

        Formating lists to slackpkg style...
                Package List
                Package descriptions
# _
```

Em seguida, deveremos utilizar...

```
# slackpkg upgrade slackware
```

O utilitário checará quais os pacotes que deverão ser atualizados através da listagem baixada pelo comando anterior. Logo em seguida solicitará confirmação para a atualização dos pacotes mencionados.

```
Looking for slackware in package list. Please, wait... DONE

espgs-8.15rc2-i486-1.tgz
flex-2.5.4a-i486-3.tgz
gaim-1.1.4-i486-1.tgz
gcc-3.3.5-i486-1.tgz
gcc-g++-3.3.5-i486-1.tgz
gcc-g77-3.3.5-i486-1.tgz
gcc-gnat-3.3.5-i486-1.tgz
```

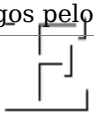
```
gcc-java-3.3.5-i486-1.tgz
gcc-objc-3.3.5-i486-1.tgz
glib2-2.6.3-i486-1.tgz
gtk+2-2.6.3-i486-1.tgz
nmap-3.81-i486-1.tgz
normalize-0.7.6-i486-1.tgz
openssh-4.0p1-i486-1.tgz
samba-3.0.11-i486-1.tgz
tetex-3.0-i486-1.tgz
tetex-doc-3.0-noarch-1.tgz
udev-054-i486-3.tgz
x11-6.8.2-i486-1.tgz
x11-devel-6.8.2-i486-1.tgz
x11-docs-6.8.2-noarch-1.tgz
x11-docs-html-6.8.2-noarch-1.tgz
x11-fonts-100dpi-6.8.2-noarch-1.tgz
x11-fonts-cyrillic-6.8.2-noarch-1.tgz
x11-fonts-misc-6.8.2-noarch-1.tgz
x11-fonts-scale-6.8.2-noarch-1.tgz
x11-xdmx-6.8.2-i486-1.tgz
x11-xnest-6.8.2-i486-1.tgz
x11-xvfb-6.8.2-i486-1.tgz

Total of package(s): 29

Do you wish to upgrade selected packages (Y/n)? _
```

É só digitar *y* + <ENTER> e aguardar o fim do processo. Em alguns casos, ele perguntará se desejamos atualizar as definições de atualização de determinados pacotes, ao serem atualizados:

```
-//-
Searching NEW configuration files

Some packages had new configuration files installed.
You have four choices:

        (K)eep the old files and consider .new files later

        (O)verwrite all old files with the new ones. The
           old files will be stored with the suffix .orig

        (R)emove all .new files

        (P)rompt K, O, R selection for every single file

What do you want (K/O/R/P)?
_
```

As opções disponíveis são:

Slackpkg	
K	Continuar com os arquivos antigos e novos, mantendo os novos como *.new* para posterior avaliação.
O	Sobrescrever os arquivos antigos pelos novos.



Slackpkg	
R	Remover todos os novos arquivos.
P	Optar por utilizar uma opção diferente para cada arquivo.

Certifiquem-se de que a listagem de espelhos aponta para um diretório *slackware-current* da distribuição. Levem em consideração o tamanho total dos pacotes à serem obtidos e a taxa de transferência da conexão.

Observações

Em algumas circunstâncias, o *Slackpkg* poderá emitir avisos como este:

```
+==============================================================================+
WARNING!        WARNING!        WARNING!        WARNING!        WARNING!
+==============================================================================+
One or more errors occurred while slackpkg was running.
One or more packages most likely could not be installed, or your mirror
is having problems. It's a good idea recheck your mirror and run slackpkg
again.
+==============================================================================+
```

Sem grandes mistérios, podemos ver que esta notificação refere-se à alguns erros que porventura possam ter ocorrido na execução desta ferramenta – especialmente na transferência dos pacotes. Como ela mesmo recomenda, verifiquem se o espelho encontra-se disponível e reexecutem o *script*.

Lembrem-se: este programa somente realiza a atualização dos pacotes pertencentes à distribuição. Para àqueles pacotes instalados de outra fonte ou compilados diretamente à partir do código-fonte, o único caminho é verificar em suas páginas oficiais se disponibilizam novas versões.

Recomendações gerais

Ao utilizarmos estas ferramentas, deveremos estar cientes de que elas podem renomear e/ou sobrescrever as configurações atuais existentes dos programas que desejamos atualizar. Nestas circunstâncias, encontraremos seus respectivos arquivos de configuração renomeados com uma nova extensão *.new* que garantirá a manutenção das definições originais, ou ainda com o sufixo ~ (<TIL> – cópia de segurança), que resguardará o arquivo original para que o mesmo seja substituído por um novo arquivo com as novas definições-padrão. Nestas circunstâncias, talvez seja necessário redefinir alguns dos parâmetros existentes para garantir o perfeito funcionamento da aplicação, onde deveremos consultar as definições do antigo arquivo para suplantá-nas no novo. Se estivermos atentos à estes detalhes, poderemos realizar intervenções corretivas de maneira mais eficaz. Em outras distribuições, poderemos até mesmo ter estes arquivos sobregravados, impedindo resgatar as definições armazenadas.

Conclusão

Diferente das demais distribuições, o *Slackware* não fornece ferramentas para atualização de pacotes; apenas disponibiliza na pasta */extra*, algumas ferramentas externas úteis para este propósito, conforme dito na introdução deste capítulo. E compilar cada pacote manualmente, ou ainda, baixar individualmente cada pacote oficial do *FTP* oficial da distribuição para a atualização do sistema, além de serem tarefas muito trabalhosas, é, dependendo das circunstâncias, praticamente inviável. Para isto, existem as ferramentas de atualização de pacotes, as quais não deveremos abrir mão.

Por isto, experimentem cada uma destas ferramentas e, de acordo com uma avaliação particular, optem por mantê-las no sistema. &;-D

III. Gerenciamento de pacotes RPM

Introdução

O *RPM – Red Hat Package* – é uma ferramenta de gerenciamento de aplicativos que consiste na administração de pacotes pré-compilados. Desenvolvido pela *Red Hat*, em sua época de lançamento o *RPM* era a grande novidade para os sistemas *GNU/Linux*, pois praticamente todos os aplicativos necessitavam de ser compilados para a sua instalação.

Dentre suas principais características, destaca-se o processo de verificação de pendências e manutenção de um banco de dados dos pacotes instalados. Ao instalar um determinado pacote no sistema, o *RPM* realiza uma consulta em seu banco de dados e verifica se os pacotes já instalados no sistema satisfazem as pendências do pacote que se deseja instalar. Caso contrário, o *RPM* emite um aviso, solicitando a instalação dos pacotes e/ou bibliotecas necessárias para a correta instalação do pacote desejado.

O *RPM* encontra-se atualmente incluso na distribuição *Slackware* devido à exigência da *LSB*, que recomenda a utilização de um gestor de pacotes universal que possibilite a instalação de diversos aplicativos.

Os comandos...

Com o *RPM*, poderemos realizar as operações de instalação, desinstalação, atualização, pesquisa, verificação e confecção. Sua sintaxe básica é:

```
# rpm [PARÂMETROS] [PACOTE]
```

As funcionalidades do *RPM* são inúmeras; veja o seu manual eletrônico:

Database maintenance:

```
rpm -i [--initdb]
rpm -i [--rebuilddb]
```

Building:

```
rpm [-b|t] [package_spec]+
rpm [--rebuild] [sourcerpm]+
rpm [--tarbuild] [tarredsource]+
```

Querying:

```
rpm [--query] [queryoptions]
rpm [--querytags]
```

Maintaining installed packages:

```
rpm [--install] [installoptions] [package_file]+
rpm [--freshen|-F] [installoptions] [package_file]+
rpm [--uninstall|-e] [uninstalloptions] [package]+
rpm [--verify|-V] [verifyoptions] [package]+
```

Signatures:

```
rpm [--verify|-V] [verifyoptions] [package]+
rpm [--resign] [package_file]+
rpm [--addsign] [package_file]+
```

Miscellaneous:

```
rpm [--showrc]
rpm [--setperms] [package]+
rpm [--setgids] [package]+
```

Estas e outras instruções encontram-se disponíveis em seu manual:

```
$ man rpm
```

À deixaremos aqui apenas para a consulta dos usuários mediano/avançados. As opções básicas são mais que suficientes para o uso cotidiano. &;-D

Instalação

A instalação de um pacote *RPM* é um processo simples e de rápida execução, em comparação ao processo de compilação do código-fonte. Um dos únicos requisitos básicos é a satisfação das pendências dos pacotes que desejamos instalar. Por exemplo, para instalar o *MPlayer*, existem uma série de outros pacotes e bibliotecas que deverão estar ou ser previamente instalados no sistema para o seu correto funcionamento. Na maioria dos casos, o *RPM* detecta estas pendências e as informa para serem sanadas.

Para evitarmos perda de tempo no processo de instalação dos pacotes que (provavelmente) necessitam da solução de pendências, visitem as páginas dos desenvolvedores e verifiquem a descrição dos pacotes necessários.

Segue abaixo um simples exemplo para a instalação de pacotes *RPM*:

```
# rpm -ivh Aplicativo-1.0-1.i386.rpm
```

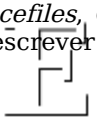
Onde:

Instalação	
-i	*Install* (instalar). Instalação do pacote desejado.
v	Modo *verbose*, exibe as informações referentes ao processo.
h	Indica o progresso da instalação, exibindo com caracter *#*.

Porém existirão circunstâncias em que será necessário a reinstalação de um determinado pacote, que por algum motivo qualquer apresenta erros. para isto existe o parâmetro *--replacepkg*, que como o próprio nome indica, se encarrega de sobrescrever o pacote já existente no sistema.

```
# rpm -ivh --replacepkg Aplicativo-1.0-1.i386.rpm
```

Existirão também casos em que a instalação dos pacotes desejados não poderão ser realizados, por motivo de conflitos entre as versões de seus respectivos arquivos e bibliotecas. Para sanar esta deficiência, poderemos lançar mão do parâmetro *--replacefiles*, que similar ao *--replacepkg*, este parâmetro se encarrega de sobrescrever apenas os arquivos já existentes



no sistema que estão apresentando conflitos.

```
# rpm -ivh --replacefiles Aplicativo-1.0-1.i386.rpm
```

Por último, para instalar "*à força*" algum pacote ao sistema, ignorando todas as suas pendências, utilize o parâmetro *--nodeps*.

```
# rpm -ivh --nodeps Aplicativo-1.0-1.i386.rpm
```

O uso deste último parâmetro no *Slackware* é bastante freqüente, face à inexistência de informações dos demais pacotes instalados em seu banco de dados. Isto deve-se ao fato de que o gerenciador de pacotes padrão do sistema é o *Pkgtool* e apesar de todas as bibliotecas estarem instaladas para um determinado aplicativo, ainda assim o *RPM* não irá detectar a sua existência no sistema.

Atualização

A atualização de um pacote *RPM* também é simples. Basta utilizarmos o seguinte comando:

```
# rpm -Uvh Aplicativo-2.0-0.i386.rpm
```

Onde:

Atualização	
-U	*Update* (atualizar). Atualiza o pacote desejado. Da mesma forma que a instalação, também faz uso dos parâmetros *v* e *h*.

Observem que o parâmetro *-U* se encontra em maiúsculo.

Caso haja algum erro proveniente da atualização de um pacote mais antigo, optem por utilizar o parâmetro *--oldpackage*.

```
# rpm -Uvh --oldpackage Aplicativo-2.0-0.i386.rpm
```

Verificação

Apesar de ser desnecessário no *Slackware*, poderemos utilizar o recurso de verificação para checar o conteúdo de um pacote ou quais pacotes contém determinados arquivos. Segue um simples exemplo:

```
# rpm -V Aplicativo
```

Onde:

Atualização	
-V	*Verify* (verificar). Pesquisa a existência de valores pré-determinados pelo usuário.
f	*File* (arquivo). Complemento para orientar o *RPM* a pesquisar qual pacote pertence um determinado arquivo.

Estes valores pré-determinados podem ser a consulta de todos os arquivos do pacote ou apenas a existência de algum específico. O exemplo citado

anteriormente apenas lista o conteúdo do pacote referenciado. Notem também a necessidade do parâmetro *-V* em maiúsculo e apenas a definição do nome do pacote que se deseja verificar. Já o comando abaixo verifica qual pacote pertence um determinado arquivo:

```
# rpm -Vf /usr/bin/vi
erro: não consigo abrir o índice de Basenames usando o db3 - Arquivo ou
diretório não encontrado (2)
o ficheiro /usr/bin/vi não pertence a nenhum pacote
# _
```

Pelo fato do *Slackware* utilizar por padrão outro gerenciador de pacotes (e com isto inexistir o banco de dados *RPM*), não será encontrado nenhuma referência ao arquivo consultado. Porém ao testarmos este comando com outras distribuições que utilizem o sistema de gerenciamento *RPM*, com certeza os resultados serão diferentes.

Consulta

Para consultar a existência de um único pacote instalado, deveremos utilizar a seguinte sintaxe:

```
# rpm -q[OPÇÕES] [PACOTE]
```

Onde:

Atualização	
-q	Verifica a existência de pacotes instalados.
a	*All* (*todos*). Refere-se a todos os dados de uma determinada classe.

Exemplo:

```
# rpm -q Aplicativo
```

Observem novamente que foi definido apenas o nome do pacote como chave de busca para a consulta do utilitário. Em casos que necessitarmos obter a listagem completa de todos os pacotes do sistema, utilizaremos...

```
# rpm -qa
```

..., onde o *RPM* fará uma consulta em seu banco de dados e mostrará na tela todos os pacotes que se encontram instalados no sistema.

Exclusão

Para que seja desinstalado um determinado pacote, deveremos utilizar...

```
# rpm -e Aplicativo
```

Onde:

Exclusão	
-e	*Exclude* (excluir). Desinstala o pacote desejado.

Da mesma forma que ocorre no processo de instalação, o *RPM* só desinstalará um determinado pacote quando não houver nenhuma dependência de outro pacote do sistema em relação ao pacote que se deseja desinstalar. Mas se ainda assim quisermos desinstalar um determinado pacote, podemos utilizar o parâmetro *–force*...

```
# rpm -e Aplicativo --force
```

... que como o próprio nome indica, "*força*" a desinstalação do pacote desejado mesmo havendo outros aplicativos que necessitam do pacote desinstalado.

Conclusão

Aprender a sintaxe dos comandos e parâmetros do gestor *RPM* é importante, pois um dos principais motivos de sua adoção está no fato deste gerenciador ser um dos requisitos exigidos pela *LSB*. Porém lembrem-se que este padrão apenas requer a manutenção deste gerenciador para garantir a possibilidade de instalação dos aplicativos empacotados em um formato universal adotado (o próprio *RPM*). Isto não quer dizer que todas as distribuição que não utilizam este gerenciador serão obrigadas à adotá-lo, ficando livres para definirem seus próprios critérios para a utilização de um formato que considerarem adequado. &;-D

IV. Compilação do código-fonte

Introdução

A compilação do código-fonte disponível de um determinado programa ou *driver*, apesar de ser menos simples que utilização de binários pré-compilados, é considerada a melhor forma de garantir a boa implementação e a excelência dos resultados que desejamos obter, de acordo com a sua finalidade. Graças à uma correta e eficiente compilação, conseguiremos alcançar excelentes níveis de otimização, performance, estabilidade, compatibilidade e ajustes altamente personalizados de que necessitamos ou quando não somente poderemos obtê-los com a utilização deste processo.

Neste capítulo iremos conhecer o processo de compilação do código-fonte, suas características, particularidades, vantagens e aplicações, como também recomendações gerais para o bom sucesso desta operação.

Considerações básicas

A compilação é o processo pelo qual as instruções contidas no código-fonte são convertidas para uma linguagem de máquina, que é somente entendida pelo próprio computador. É muitas vezes também chamada de código "*assemblado*". Para os usuários mais novatos, vindos do sistema operacional *Windows*, muitos destes sentem-se confusos sobre o processo de compilação, pois os mesmos estão habituados à lidarem com a instalação de programas à partir do pacote ou um conjunto de arquivos binários, os quais contém suas aplicações prediletas.

De acordo com as necessidades (e vantagens), existem duas formas básicas de compilação para um programa: A estática e a dinâmica.

A compilação estática é um processo que apresenta o conceito de integração total do programa, com todos os seus arquivos e bibliotecas necessárias para o seu funcionamento. A principal vantagem está no fato de que não haverá necessidade de instalação de bibliotecas necessárias para o perfeito funcionamento do programa – as conhecidas pendências. Em contrapartida, o tamanho ocupado pelo programa é muito superior à utilização destes mesmos utilizando o processo de compilação dinâmica, além de ocupar muito mais memórias, caso dois ou três destes programas utilizem as mesmas bibliotecas e estejam em execução ao mesmo tempo.

A compilação dinâmica apresenta conceitos *"inversos"* da estática: ao invés de incluir as bibliotecas necessárias no corpo principal do programa, estas são instaladas à parte, onde o programa principal somente realiza as chamadas de funções necessárias para a execução de suas atividades. A grande vantagem deste método de compilação está justamente nas desvantagens do método de compilação estática: ganha-se na confecção de

pacotes enxutos, utilização de pouca memória e ótima performance quando suas bibliotecas necessárias sendo utilizadas pelo ambiente, como é o caso das bibliotecas gráficas *Qt* e *GTK*. Em contrapartida, surge o grande inconveniente das pendências, ou seja, para a instalação destes programas, serão necessários a instalação das devidas bibliotecas para a sua utilização, caso elas não constem no sistema.

Dentre os principais programas que utilizam o recurso de compilação estática encontram-se todos os aplicativos disponíveis para a plataforma *Windows*, além do *Mozilla*, *Kylix* e *OpenOffice.org* para os sistemas *GNU/Linux*, levando em consideração a existência de inúmeros programas comerciais portados. Este é o principal motivo do qual a instalação de programas para *Windows* não requer a instalação de pendências de pacotes, à salvo quando há a necessidade de utilização de *plugins*.

Felizmente na instalação das tradicionais distribuições, grande parte das bibliotecas também se encontram instaladas, bastando apenas resolver algumas pendências específicas pelo programa utilizado. Este é o principal motivo pelo qual dezenas de programas cabem em apenas um único *CD-ROM* de instalação desta distribuição, já que utilizam este conceito.

As ferramentas do Projeto GNU

Conforme dissemos anteriormente na *1a. Parte: Os sistemas GNU/Linux -> Linux*, o *Projeto GNU* consistia de desenvolver um sistema operacional *Unix* livre, onde para isto foram construídas inicialmente diversas ferramentas de desenvolvimento. Dentre elas existem desde compiladores, editores, utilitários de automação e uma série de bibliotecas para esta atividade. Iremos conhecer as ferramentas indispensáveis para a realização do processo de compilação.

Bibliotecas

GNU C Library

- ✔ <http://www.gnu.org/software/libc/libc.html>.

A *GNU C Library* – conhecida como *glibc* – é usada por praticamente todos os programas desenvolvidos para os sistemas *GNU/Linux*. Ela é essencial para a compilação dos códigos-fonte de seus pacotes.

Todas as distribuições possuem uma versão da *glibc* instalada no sistema, onde de acordo com a filosofia de cada uma, poderemos ter a versão mais recentes ou não. Na consulta das instruções *README* e *INSTALL* dos pacotes com os fontes, apenas deveremos ficar atento às versões exigidas da *glibc*, onde raramente as distros deixam de atender tais requisitos.

GNU Libtool

- ✔ <http://www.gnu.org/software/libtool/>.

Desenvolvido por *Gordon Matzigkeit*, o *GNU Libtool* é um conjunto de *shell scripts* desenvolvidos para facilitar a criação e uso de bibliotecas compartilhadas. Sua presença é indispensável para a compilação de bibliotecas e pendências necessários para a instalação de outros programas.

Ferramentas de automação

AUTOCONF

- ✔ <http://www.gnu.org/software/autoconf/>.

O *autoconf* é uma ferramenta que tem por objetivo desenvolver *scripts* para automatizar a configuração de códigos-fontes para a compilação em diferentes plataformas. Graças à ele torna-se possível utilizar um mesmo código para diferentes arquiteturas. Com ele é criado o *script configure*.

O *script configure*, por sua vez, examina as variáveis, a existência de pendências e as especificações da plataforma. Com estas informações, ele irá construir os arquivos *Makefiles* personalizados, que serão utilizados como regras para a compilação dos programas e assim garantir a compilação do programa específicamente para o sistema em uso.

m4

- ✔ <http://www.gnu.org/software/m4/>.

O *m4* é a implementação de macro-processadores tradicionais dos ambientes *Unix*. Sua função consiste em gerar os arquivos *Makefile.am*, que por sua vez contém as instruções necessárias para que o *automake* possa construir os *Makefile.in*.

AUTOMAKE

- ✔ <http://www.gnu.org/software/automake/>.

O *automake* é um *script* desenvolvido em *Perl* que utiliza as definições geradas pelo *GNU m4* (*Makefile.am*) para a construção dos arquivos *Makefile.in*. Sua utilização traz a vantagem de automatizar a tarefa de manutenção dos *Makefiles*.

MAKE

- ✔ <http://www.gnu.org/software/make/make.html>.

Quando se compila um programa, não só existe a necessidade de unir (*link*) todo o código-fonte e gerar o programa principal, como também incluir

diversas outras instruções (*bibliotecas*) para compor o corpo funcional do programa. Realizar esta operação é algo extremamente trabalhoso, pois teríamos que realizar diversos procedimentos manuais para satisfazer suas necessidades. Para isto existe o comando *make*.

O comando *make* realiza a compilação propriamente dita dos programas os quais desejamos instalar, onde este segue as instruções de um arquivo especial gerado pelo *configure* – os *Makefiles* –, os quais contém todas as instruções para a realização do processo.

Compiladores

Existem diversos compiladores desenvolvidos pelo *Projeto GNU*, onde o mais famoso (e largamente utilizado) é o *GNU C Compiler*.

GNU C Compiler

- ✔ *<http://gcc.gnu.org/>.*

Conhecido popularmente como *GCC*, é de longe o compilador mais utilizado pelos sistemas *GNU/Linux*, como também no *Windows*. Desenvolvido por *Richard Stallman* com o apoio de voluntários, o *GCC* é um dos melhores e mais completos compiladores existentes. Suporta *C*, *C++*[1] e *Objetive-C*, segue as especificações *ANSI C*, possui excelente performance e portabilidade, além de inúmeras outras qualidades.

O *GCC* encontra-se disponível por padrão em praticamente todas as distribuições *GNU/Linux*. Em algumas existirão compiladores condicionados para serem utilizados especificamente em algumas arquiteturas específicas, como o *Pentium* da *Intel*, por exemplo. Ele é indispensável para a compilação da grande maioria dos programas desenvolvidos para sistemas *GNU/Linux*, por serem muitas vezes desenvolvidos em *C*.

Diversos

binutils

- ✔ *<http://www.gnu.org/software/binutils/>.*

Este pacote contém um conjunto de ferramentas para a manipulação de arquivos binários nos sistemas *GNU/Linux*. Na compilação dos programas, é utilizado para a construção dos arquivos que irão compor o corpo do programa em si, inclusive o executável.

1 Antigamente o *GCC* possuía uma pequena limitação de desempenho ao compilar programas escritos em *C++*, ficando atrás de compiladores comerciais famosos desenvolvidos pela *Intel* e *Borland*. Porém, à partir da versão *3.1*, foram acrescentadas diversas melhorias os quais tornaram esta limitação uma simples lembrança do passado ;-D

Patch

- ✔ <http://www.gnu.org/software/patch/patch.html>.

O comando *patch* é utilizado para atualizar um "*antigo*" programa empacotado no formato de código-fonte, que por sua vez necessita de uma atualização. Os motivos desta atualização poderão ser diversos: correção de erros, atualização propriamente dita, compatibilidade, etc. Os pacotes de atualização possuem a extensão *.diff*, os quais poderão ser baixados normalmente como um arquivo comum.

Antes de utilizar o pacote com a atualização, deveremos copiá-lo para o diretório-raíz onde se encontra o código-fonte do programa original e dentro deste, utilizar as seguintes sintaxes:

Para atualizar um pacote, utilize...

```
$ patch -p0 < [ATUALIZAÇÃO].diff
```

Para testar se a operação foi realizada normalmente...

```
$ patch -p0 --dry-run [ATUALIZAÇÃO].diff
```

Para remover a atualização...

```
$ patch -p0 -R < [ATUALIZAÇÃO].diff
```

A compilação padrão

A compilação padrão – que também chamamos de compilação clássica – é a forma mais simples e básica utilizada para realizar a compilação de um programa. O processo é relativamente fácil, onde não existe a necessidade de mais nenhum outro parâmetro ou instrução adicional para que se possa instalar o programa desejado.

Segue abaixo a descrição de um processo genérico para a compilação de código-fontes de programas para as mais diversas finalidades, passando por aplicativos, utilitários, *drivers*, bibliotecas, *APIs*, etc.

Obtenção do código-fonte

Geralmente você obtém o código-fonte dos programas ou *drivers* desejados na página do desenvolvedor. Basta apenas baixá-los e copiá-los para a máquina onde você deseja instalar ou obtê-los de diferentes locais, como nos *CD-ROMs* que possuem o código-fonte dos aplicativos e outros dispositivos de armazenamento (cópia de segurança).

Descompactação do código-fonte

O código-fonte geralmente se encontra empacotado e compactado em um único arquivo que pode ter a extensão *.tar.gz* ou *.tar.bz2*. A *1a.* extensão *.tar* informa que o código fonte encontra-se empacotado pelo utilitário *TAR*, enquanto o complemento informa que o pacote gerado foi compactado com

o *gzip* (*.tar.gz*) ou *Bzip2* (*.tar.bz2*). para descompacta-los, abra um terminal e proceda da seguinte forma.

- Para pacotes compactado com o *gzip* (extensão *.tar.gz*)...

```
$ tar -xpzvf [NOME DO PACOTE].tar.gz
```

... ou...

```
$ gunzip [NOME DO PACOTE].tar.gz
$ tar -xpvf [NOME DO PACOTE].tar
```

- Para pacotes compilados com o *Bzip2* (extensão *.tar.bz2*)...

```
$ tar -xpjvf [NOME DO PACOTE].tar.bz2
```

... ou...

```
$ bunzip2 [NOME DO PACOTE].tar.bz2
$ tar -xpvf [NOME DO PACOTE].tar
```

Estes são os compactadores mais comuns utilizados no sistema *GNU/Linux*, porém existe a possibilidade de se obter o pacote do código-fonte compactado com outros utilitários menos conhecidos:

- Para pacotes compactados com o *compress* (extensão *.tar.Z*)...

```
$ tar -xpZvf [NOME DO PACOTE].tar.Z
```

... ou...

```
$ compress -d [NOME DO ARQUIVO].tar.Z
$ tar -xpvf [NOME DO ARQUIVO].tar
```

Observem a utilização do parâmetro *Z* em maiúsculo, informando ao empacotador que o arquivo à ser desempacotado foi compactado pelo utilitário *compress*.

- Para pacotes compactados no formato *ZIP* (extensão *.zip*):

```
$ unzip [NOME DO PACOTE].zip
```

Após a realização dos passos demonstrados, normalmente é criado um diretório com o nome do programa e o conteúdo do código-fonte desempacotado. Agora basta apenas entrar neste diretório...

```
$ cd [NOME DO DIRETÓRIO CRIADO]
```

Local de armazenamento

Conforme a recomendação técnica da norma *FHS*, os pacotes que contém o código-fonte dos programas deverão estar armazenados no diretório */usr/local/src* – pois não pertencem à distribuição –, mas nada o impede de armazená-lo em qualquer outro lugar. Se desejarmos, poderemos armazená-los no diretório de usuário (em nosso caso, */home/darkstar*) para nosso uso e administração particular.

Leitura de documentações

Ao consultarmos a estrutura do diretório criado ao descompactar um

determinado pacote, iremos observar uma estrutura de dados contendo vários arquivos e diretórios.

```
# ls -l
total 1987
-rw-r--r--    1 darkstar users      353197 2004-02-02 18:49 acinclude.m4
-rw-r--r--    1 darkstar users      383288 2004-02-02 18:49 aclocal.m4
drwxr-xr-x    2 darkstar users         864 2004-02-02 18:22 admin
-rw-r--r--    1 darkstar users        2600 2004-02-02 18:49 AUTHORS
-rw-r--r--    1 darkstar users       10196 2004-02-02 18:43 ChangeLog
-rw-r--r--    1 darkstar users        9793 2004-02-02 18:50 config.h.in
-rwxr-xr-x    1 darkstar users     1160095 2004-02-02 18:50 configure
-rw-r--r--    1 darkstar users         274 2004-02-02 18:49 configure.files
-rw-r--r--    1 darkstar users       24162 2004-02-02 18:49 configure.in
-rw-r--r--    1 darkstar users        3699 2004-02-02 18:22 configure.in.in
-rw-r--r--    1 darkstar users         776 2004-02-02 18:07 COPYING
-rw-r--r--    1 darkstar users       10369 2004-02-02 18:07 INSTALL
drwxr-xr-x   11 darkstar users         544 2004-02-02 20:00 kopete
-rw-r--r--    1 darkstar users         241 2004-02-02 18:49 Makefile.am
-rw-r--r--    1 darkstar users         215 2004-02-02 18:22 Makefile.am.in
-rw-r--r--    1 darkstar users       22123 2004-02-02 18:50 Makefile.in
drwxr-xr-x   39 darkstar users        1000 2004-02-02 20:00 po
-rw-r--r--    1 darkstar users        1326 2004-02-02 18:07 README
-rw-r--r--    1 darkstar users           0 2004-02-02 18:50 stamp-h.in
-rw-r--r--    1 darkstar users          10 2004-02-02 18:49 subdirs
-rw-r--r--    1 darkstar users        3898 2004-02-02 18:07 TODO
-rw-r--r--    1 darkstar users          13 2004-02-02 18:40 VERSION
# _
```

Estrutura dos arquivos-fontes do Kopete, um gerenciador de mensagens instantâneas

De acordo com o pacote instalado, veremos inúmeros ítens disponibilizados, porém todos pacotes deverão ter em comum um sub-diretório com o nome docs ou pelo menos no diretório raiz deverão constar os arquivos *README*, *INSTALL* e *COPYING*. Estas são as informações necessárias escritas em formato texto ou html com as instruções necessárias para o correto uso do código-fonte, além de seu licenciamento.

Além destes arquivos, poderão existir diversos outros inclusos no diretório sobre a utilização do código-fonte em questão. Procurem sempre consultá-los antes de realizar qualquer operação, pois apesar de se encontrarem em inglês, com certeza todas as suas possíveis particularidades estarão lá descritos, além dos respectivos procedimentos, os quais serão muito importantes para que se possa obter um melhor êxito na manipulação dos programas desejados.

A CONFIGURAÇÃO, COMPILAÇÃO E INSTALAÇÃO DO PACOTE

Para compilar o pacote, existem *3* comandos à serem utilizados:

Comandos & Funções	
./configure	*Script* responsável para detecção das pendências e geração de relatórios, chamados *makefile*.

Comandos & Funções	
make	Ferramenta de automação necessária para a compilação do código-fonte, baseando-se nas instruções do *makefile*.
make install	Instalação do pacote compilado.

A partir daí, é só digitar os seguintes comandos e aguardar algumas instruções do processo de compilação, de acordo com o pacote.

```
# ./configure
-//-
# make
-//-
# make install
```

Os comandos *./configure* e *make* poderão ser rodados com permissões de usuários comum; já o *make install* somente poderá ser utilizado com os privilégios do superusuário, pois é justamente ele que irá realizar a instalação propriamente dita.

Limpando o diretório do pacote compilado

Após a realização da compilação e instalação de um programa, talvez seja necessária a limpeza da estrutura do diretório com o código-fonte. Para isto temos à disposição a opção *make clean* para *limpar* (apagar) os arquivos gerados durante o processo de compilação, bem como as definições gerais utilizadas durante a execução do *script* de configuração.

```
# make clean
```

Este processo somente se faz necessário quando houver necessidade de reutilizar o código-fonte para realizar novas compilações ou caso tenha ocorrido algum erro em alguma compilação prévia (e depois solucionada), de acordo com as nossas necessidades.

Desinstalando um pacote compilado

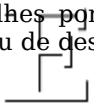
Alguns programas possuem a opção *make uninstall*, necessário para desinstalar os programas compilados no sistema.

```
# make uninstall
```

Recomenda-se a utilização de utilitários para o gerenciamento de programas compilados no sistema, como o *Checkinstall*.

Considerações avançadas

Até agora, nós observamos apenas uma demonstração genérica de um simples processo de compilação de um aplicativo ou utilitário. A partir desta seção, teremos uma visão mais ampla e minuciosa acerca do assunto, onde analisaremos diversos detalhes pormenores dos quais muitas vezes são tidos como sem importância ou de desnecessária atenção.



Conteúdo da documentação

Ao contrário do que muitos pensam, uma das mais importantes etapas do processo de compilação é a leitura e o entendimento de toda sua documentação, ou pelo menos das partes relevantes do texto.

Como requerimento básico, todos os desenvolvedores de programas devem redigir uma documentação de sua aplicação, o qual deverá descrever todos os processos básicos inerentes como os requerimentos básicos, a instalação, a configuração, a utilização, diversas recomendações e muitas outras informações necessárias para o bom uso do programa.

O diretório docs

Geralmente a maioria dos desenvolvedores quando não disponibilizam a documentação em um arquivo separado ou quando estes possuem um certo grau de complexidade, geralmente armazenam diversos documentos (arquivos) em um subdiretório chamado *docs*. Estarão lá as principais informações pertinentes ao programa em questão.

Readme / Install

Instruções básicas sobre o programa e o procedimento correto para a instalação. Apesar de muitas vezes ser descartado, é imprescindível a sua leitura, pois muitas dos problemas e dificuldades encontrados estão brevemente descritos nestes documentos. Muitas vezes são inclusos os dois arquivos distintos: o *README* para as instruções gerais...

```
# cat README
                          fetchmail README

Fetchmail is a free, full-featured, robust, well-documented remote
mail retrieval and forwarding utility intended to be used over
on-demand TCP/IP links (such as SLIP or PPP connections).  It
retrieves mail from remote mail servers and forwards it to your local
(client) machine's delivery system, so it can then be be read by
normal mail user agents such as elm(1) or Mail(1).

Fetchmail supports all standard mail-retrieval protocols in use on the
Internet: POP2, POP3, RPOP, APOP, KPOP, IMAP2bis, IMAP4, IMAP4rev1
ESMTP ETRN, and ODMR.  Fetchmail also fully supports authentication
via GSSAPI, Kerberos 4 and 5, RFC1938 one-time passwords, Compuserve's
POP3 with RPA, Microsoft's NTLM, Demon Internet's SDPS, or CRAM-MD5
authentication a la RFC2195.  Fetchmail also supports end-to-end
encryption with OpenSSL.

The fetchmail code was developed under Linux, but has also been
extensively tested under the BSD variants, AIX, HP-UX versions 9 and
10, SunOS, Solaris, NEXTSTEP, OSF 3.2, IRIX, and Rhapsody.

It should be readily portable to other Unix variants (it uses GNU
autoconf).  It has been ported to LynxOS and BeOS and will build there
without special action.  It has also been ported to QNX; to build
-//-
```

... e o *INSTALL* para as instruções com enfoque exclusivo para a instalação.

```
$ cat INSTALL
                INSTALL Instructions for fetchmail

If you have installed binaries (e.g. from an RPM) you can skip to step 5.

If you are a Linux system packager, be aware that the build process
generates
an RPM spec file at fetchmail.spec, and you can "make rpm" to generate an
RPM and SRPM.

The Frequently Asked Questions list, included as the file FAQ in this
distributions, answers the most common questions about configuring and
running fetchmail.

1. USEFUL THINGS TO INSTALL FIRST

If you want support for RFC1938-compliant one-time passwords, you'll
need to install Craig Metz's OPIE libraries first and *make sure
they're on the normal library path* where configure will find them.  Then
configure with --enable-OPIE, and fetchmail build process will detect
them and compile appropriately.

Note: there is no point in doing this unless your server is
OTP-enabled.  To test this, telnet to the server port and give it
a valid USER id.  If the OK response includes the string "otp-",
you should install OPIE.  You need version 2.32 or better.
-//-
```

Em muitas circunstâncias, estas instruções poderão se encontrar bastante resumidas. Veja as instruções do *INSTALL* do *XChat*:

```
$ cat INSTALL
X-Chat Requirements:
~~~~~~~~~~~~~~~~~~~~

        - GTK 2.0 or 2.2  (get it from http://www.gtk.org)

 Optional:

        - Perl                (http://www.perl.org)
        - Python              (http://www.python.org)
        - TCL                 (http://tcl.activestate.com)
        - OpenSSL             (http://www.openssl.org)



X-Chat Compiling and Installation:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Type this:

        ./configure
        make

Become root and type:

        make install
```

```
Other Options
~~~~~~~~~~~~~

To get a full list of compile options, type:

        ./configure --help
-//-
```

Copying

O documento *COPYING* contém a licença do programa, suas cláusulas, termos, restrições, entre outros. Nos programas de código-aberto distribuídos livremente, geralmente se encontra uma licença *GPL* ou compatível. Para ter acesso ao seu conteúdo, basta utilizarmos qualquer aplicação para a sua leitura.

```
$ cat COPYING

                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.


                           675 Mass Ave, Cambridge, MA 02139, USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                              Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.
-//-
```

As cláusulas redigidas nesta licença deverão ser as mesmas, mesmo que estas se encontrem em diferentes formatações. Porém em algumas circunstâncias, encontraremos estas instruções resumidas em outros arquivos que tratam sobre o licenciamento. Vejam o exemplo obtido na documentação do compactador *Bzip2*:

```
$ cat LICENSE

This program, "bzip2" and associated library "libbzip2", are
copyright (C) 1996-2002 Julian R Seward.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

```
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. The origin of this software must not be misrepresented; you must
   not claim that you wrote the original software.  If you use this
   software in a product, an acknowledgment in the product
   documentation would be appreciated but is not required.

3. Altered source versions must be plainly marked as such, and must
   not be misrepresented as being the original software.

4. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, Cambridge, UK.
jseward@acm.org
bzip2/libbzip2 version 1.0.2 of 30 December 2001
```

Conforme dito, apesar do corpo da licença não se encontrar, estão descritas as principais cláusulas necessárias para torná-la compatível com a *GPL*.

Copyright

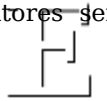
O documento *COPYRIGHT* contém trechos sobre o licenciamento de partes ou aspectos importantes do programa. Nele poderemos consultar para saber se uma aplicação é disponível sob os termos da *GNU GPL*, além de encontrar o nome dos criadores do produto.

Release

Todas as implementações, melhorias e correções do programa são especificados neste documento. Quando houver necessidade de se consultar se um determinado programa suporta uma tecnologia ou inovação, é neste arquivo em que deverão ser consultadas estas informações.

Credits

Todos os devidos créditos, que vão desde autores à colaboradores, suportes, etc., estão especificados neste documento. Apesar de ser dispensável sua leitura, seus autores sentirão-se lisongeados em serem conhecidos... &;-D



Changelog

As principais mudanças realizadas ao longo da existência do programa são comentados brevemente neste documento, que é praticamente um histórico de todas as versões de sua existência. Poderemos encontrar nestas documentações as principais implementações, além de correções de erros e ainda os principais motivos do porque um programa *X* não funciona com a versão da biblioteca *Y*, entre outros. Segue um pequeno trecho do *Changelog* do *HotPlug*.

```
$ cat Changelog
Tue Aug 5 2003 kroah
        - 2003_08_05 release

Sun Aug 3 2003 kroah
        - fix usb autoloading of modules for 2.6 (it wasn't working).

Fri Jun 27 2003 kroah
        - changed network code to accept 2.5's change to the way network
          interfaces are brought up and down (now "add" and "remove" like all
          the other hotplug types.)

Fri Jun 6 2003 dbrownell
    hotplug.functions fixes from:
        - Olaf Hering:  #!/bin/bash gone, VIM modeline,
          logger location can vary, and should include PID
        - Ian Abbot: correct init of LOADED (fixes sf.net bug filing)
    Other sf.net bugs resolved
        - pointless pci.rc message [538243]
        - Makefile should ignore tape drives [578462]

Thu May 1 2003 kroah
        - 2003_05_01 release
        - made /sbin/hotplug a tiny multiplexer program, moving the original
          /sbin/hotplug program to /etc/hotplug.d/default/default.hotplug
-//-
```

Funcionalidades detalhadas

Apesar de serem necessárias apenas a utilização da seqüência de comandos padrões para a compilação clássica na maioria das circunstâncias, existirão casos em que, para instalarmos determinados aplicativos, teremos que recorrer à utilização de intervenções específicas para propósitos distintos; seja para habilitar um suporte ou uma funcionalidade extra, seja para adaptar o programa de acordo com as nossas necessidades. Neste caso, a seqüencia de comandos muito provavelmente será modificada para atender à tais circunstâncias; ora existirão outros comandos à mais, ora os mesmos comandos necessitarão de um parâmetro extra.

Dos comandos e parâmetros geralmente mais utilizados, seguem:

./configure

Conforme dissemos anteriormente, o *script configure* é o responsável pela construção dos *Makefiles* para que o compilador construa os binários à

partir de suas especificações.

```
# ./configure [PARÂMETROS]
```

Apesar da maioria dos programas utilizarem esta opção padrão para a configuração dos pacotes à serem compilados, muitos possuem parâmetros específicos que deverão ser habilitados e/ou desabilitados na própria linha de comando. Por exemplo, o *MPlayer* é um dos programas que necessitam de diversos parâmetros especificados nesta linha.

Para obter maiores informações dos parâmetros disponíveis, consultem as instruções contidas nos arquivos *README* ou *INSTALL*, ou utilizem o parâmetro *--help* na própria linha de comando. Segue um pequeno exemplo das opções disponíveis por padrão:

```
# ./configure --help
`configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']

-//-
```

Outra situação bastante comum é o interesse do usuário em instalar o programa em locais específicos. Para isto bastará a adição do parâmetro...

```
$ ./configure --prefix=/[LOCALIZAÇÃO]
```

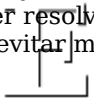
Por padrão, os programas compilados são instalados na árvore */usr/local*, porém sua localização pode ser modificada conforme as especificações do parâmetro *--prefix=*, onde *[LOCALIZAÇÃO]* será o novo diretório destino.

MAKE

O comando *make* é o mais simples de utilizar, pois não necessita de nenhum parâmetro extra para a sua utilização – basta apenas digitar na linha de comando...

```
$ make
```

... para iniciar todo o processo de compilação. Em contrapartida, caso ocorra algum erro durante a utilização deste comando, existirá uma grande impossibilidade deste problema ser resolvido. Procurem sempre consultar a documentação do programa para evitar maiores inconvenientes.



MAKE DEPS / MAKE DEPEND

Estes comandos possuem a finalidade de checar as pendências gerais do sistema para a satisfação do programa que se deseja instalar. Caso falte algum pacote ou modificação necessária, estas informações serão exibidas no programa. São poucos, mas alguns programas necessitam destes parâmetros.

MAKE INSTALL

Diferente dos comandos anteriores, o comando *make install* somente é executado pelo superusuário, e é o principal responsável pela correta instalação do programa. Somente poderá ser utilizado depois de se ter obtido sucesso com as etapas anteriores. Sem grandes mistérios.

MAKE CLEAN

Conforme dito anteriormente, remove ("*limpa*") todos os arquivos gerados pela compilação anterior e que se mantiveram presentes na estrutura de diretórios do código-fonte.

MAKE UNINSTALL

Sua função é desinstalar os programas compilados (inverso de *make install*), porém com um pequeno agravante: nem todos os programas possuem esta opção para realizar a sua desinstalação.

A melhor forma de suprir esta deficiência é a execução de utilitários para o gerenciamento de programas, como o *Checkinstall* ao invés de utilizar o comando *make install*. Será então gerado um pacote compilado no formato desejado, que possibilitará utilizar as ferramentas de gerenciamento de pacotes do sistema e com isto realizar sua remoção tranqüilamente.

Antes de realizar qualquer instalação, certifique-se de que a opção *make uninstall* encontra-se disponível no programa à ser compilado, para evitar este tipo de inconveniente.

OBSERVAÇÕES IMPORTANTES PARA O PROCESSO DE COMPILAÇÃO

MANUTENÇÃO DO CÓDIGO-FONTE

À salvo algumas situações especiais, o diretório com o código-fonte do aplicativo após devidamente compilado e instalado, poderá ser removido. O espaço utilizado pelos arquivos gerados no processo de compilação tende à ser grande, ocupando desde vários à centenas de *megabytes*. Caso tenham que intervir novamente no gerenciamento do programa instalado, reutilizem as opções disponíveis do código-fonte empacotado ou as ferramentas nativas do sistema para gerenciamento de programas.

Aplicativos & utilitários

Praticamente todos os aplicativos e utilitários possuem requerimentos de bibliotecas e *APIs* para serem corretamente instalados, e para obter informações sobre eles, consulte sua página eletrônica ou a documentação que acompanha o código-fonte do aplicativo.

Outro grande inconveniente está nas versões das pendências e bibliotecas necessárias. Muitas vezes os pacotes que compõe a distribuição se encontram desatualizados para os aplicativos que desejamos instalar, sendo necessário a atualização destes para a obtenção de bons resultados.

Drivers, módulos e kernel

Dentre as necessidades mais importantes para a compilação de *drivers* e módulos está na manutenção do código-fonte do *kernel* (e em alguns casos, do cabeçalho) previamente instalado. Procurem se certificar de que os pacotes *kernel-headers* e *kernel-source* se encontram instalados no sistema. Dependendo da distribuição em uso, estes pacotes não se encontram no *CD-ROM* de instalação, onde deverão ser baixados e instalados.

Para o carregamento e descarregamento dos módulos compilados, será indispensável a manutenção do pacote *modutils*, que felizmente se encontra em todas as distribuições *GNU/Linux*, além de ser instalado por padrão.

Diversos periféricos necessitam da criação, edição e modificações na permissão de acesso de seus respectivos dispositivos, como é o caso das placas de som, *fax-modem*, *CD-R/CD-RW*, leitores de *DVD*, etc.

Com menor incidência, existe a necessidade da edição de parâmetros em arquivos de configuração, tanto o do sistema como o do próprio programa.

Por último, o processo de compilação do *kernel* de sistemas *GNU/Linux* é bem diferenciado, em comparação ao processo de compilação dos programas e *drivers*. Além da existência de inúmeros parâmetros, existem uma série de requerimentos necessários para que possamos garantir a obtenção de excelentes resultados. Para obter maiores informações, consultem a *8a. Parte*: *O Kernel e a Compilação*.

Conclusão

Conforme dito na introdução deste capítulo, o processo de compilação, se não é a única, é a melhor forma de garantir uma perfeita interação das aplicações com o sistema operacional. Com a utilização de ajustes, parâmetros e personalizações extra, não somente teremos garantido o seu perfeito funcionamento, como também a melhoria de sua performance com técnicas e otimização, utilização de recursos extras, entre outros artifícios os quais se façam necessários.

Freqüentemente iremos nos deparar com situações em que teremos a necessidade de usar este processo em situações diversas, em especial para

a instalação de *drivers* e implementação de outros (ou novos) recursos ao sistema e suas aplicações. Para isto é de extrema importância que devemos conhecer os fundamentos desta "*arte*", que por sua vez torna os sistemas *GNU/Linux* imbatíveis em termos de performance e flexibilidade! &;-D

V. Conversão de pacotes e programas

Introdução

São diversos os motivos os quais levam os usuários à realizar conversões de pacotes e programas compilados para o formato nativo do *Slackware* – o principal deles está na ausência do pacote desejado, lógico! Neste capítulo, teremos algumas instruções e recomendações necessárias para realizar esta atividade com alguns dos principais utilitários para esta função.

As ferramentas...

Checkinstall

- ✔ <http://asic-linux.com.mx/~izto/checkinstall/>.

O *Checkinstall* foi desenvolvido por *Felipe Eduardo Sánchez Díaz Durán* e trata-se de um utilitário desenvolvido especialmente para empacotar programas compilados manualmente pelo administrador do sistema.

Muitas vezes, quando instalamos programas à partir do código-fonte, ficamos sem nenhuma condição de gerenciá-lo, pois em muitos destes as ferramentas de manipulação – exclusão, mais especificamente – não existem. E é justamente nestas circunstâncias onde entra o *Checkinstall*...

A instalação

O *Checkinstall* se encontra disponível no *CD-ROM Extra* da distribuição à partir da versão *9.0*. Instale o pacote lá disponível ou vá ao *FTP* oficial do *Slackware* e, na pasta */extra*, obtenha o pacote e instale com o comando...

```
# installpkg checkinstall-[VERSÃO].tgz
```

Poderemos também obter o pacote do autor disponível na página oficial do projeto. Lá encontraremos disponível tanto uma versão pré-compilada para esta e outras distribuições quanto o pacote com o código-fonte.

Para instalar o pacote que contém o código-fonte, deveremos utilizar...

```
# make
# make install
```

Por último...

```
# checkinstall -y -S
```

... para que a própria instalação do *Checkinstall* possa ser gerenciado pelo sistema nativo de gerenciamento de pacotes. Por fim, após responder todas as perguntas do utilitário, conclua a "instalação" do programa com...

```
# installpkg checkinstall-[VERSÃO-ARQUITETURA].tgz
```

Assim o *Checkinstall* estará devidamente empacotado e instalado no

sistema. Afinal de contas, se é para auxiliar o gestor de pacotes, porquê não começar à dar o bom exemplo? &;-D

A UTILIZAÇÃO

Para utilizar o *Checkinstall*, bastará apenas configurar e compilar o código-fonte de qualquer programa com a utilização dos comandos...

```
# ./configure [OPÇÕES]
# make
```

..., porém ao invés de realizar a instalação com o comando *make install*, bastará digitar na linha de comando...

```
# checkinstall
```

...e o *script* lhe criará o pacote desejado automaticamente, baseado na respostas de algumas instruções feitas pelo próprio utilitário. Caso queiramos criar um pacote sem ter que responder as perguntas feitas, deveremos utilizá-lo com o parâmetro *-y*:

```
# checkinstall -y
```

Com este comando, o aplicativo aceitará as respostas como positivo e criará um novo pacote personalizado, porém solicitando informar o formato de pacotes à utilizar (*S – Slackware*, *D – Debian*, *R – RPM*). Mas se o utilizarmos com a opção *-S*...

```
# checkinstall -y -S
```

..., os binários do programa serão automaticamente empacotados no formato *.tgz* sem qualquer questionamento. O mesmo processo vale para os demais formatos, sendo *-D* para o formato *.deb* e *-R* para o formato *.rpm*.

Segue abaixo alguns exemplos de utilização, utilizando um aplicativo fictício chamado *Aplicativo* como exemplo. Conforme as instruções acima, ao invés de digitar *make install* + <ENTER>, digitaremos apenas *checkinstall* + <ENTER>. Logo em seguida o utilitário apresentará uma série de instruções para criarmos o pacote compilado.

```
# checkinstall
checkinstall 1.5.3, Copyright 2001 Felipe Eduardo Sanchez Diaz Duran
            This software is released under the GNU GPL.


The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs?  [y]:
```

Aqui somos questionados se desejamos criar o pacote com a sua documentação. Em caso negativo teclem *n* + <ENTER>. Como pretendemos adicionar a documentação, deveremos teclar <ENTER>.

```
Preparing package documentation...OK

*** No known documentation files were found. The new package
*** won't include a documentation directory.

Installing with "make install"...
```

```
======================== Installation results ==========================
-//-
======================== Installation succesful ========================

Some of the files created by the installation are inside the build
directory: /usr/local/src/Aplicativo-1.0.0

You probably don't want them to be included in the package,
especially if they are inside your home directory.
Do you want me to list them?  [n]:
```

Nesta etapa o utilitário pergunta se desejará verificar o conteúdo do pacote que será criado (listagem). Seguindo novamente as opções padrões do programa, teclem em <ENTER>.

```
Should I exclude them from the package? (Saying yes is a good idea)  [y]:
```

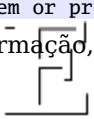
Já nesta parte, o utilitário questiona se deveremos criar um pacote com o programa compilado (para depois reinstalá-lo sem recompilá-lo novamente). Confirme sua recomendação, teclando apenas <ENTER>.

```
Copying files to the temporary directory...OK

Striping ELF binaries and libraries...OK

Compressing man pages...OK

Building file list...OK


Please write a description for the package. Remember that pkgtool shows
only the first one when listing packages so make that one descriptive.
End your description with an empty line or EOF.
>> _
```

Insiram as informações gerais sobre o pacote que está compilando. Após terminar, teclem <ENTER> em uma linha vazia e aguardem.

```
>> Aplicativo compilado manualmente.
>> 25/12/2003.
>> <ENTER>


This package will be built according to these values:

1 -  Summary: [ Aplicativo compilado manualmente ]
2 -  Name:    [ Aplicativo ]
3 -  Version: [ 1.0.0 ]
4 -  Release: [ 1 ]
5 -  License: [ GPL ]
6 -  Group:   [ Applications/System ]
7 -  Architecture: [ i386 ]
8 -  Source location: [ Aplicativo-1.0.0 ]
9 -  Alternate source location: [  ]

Enter a number to change any of them or press ENTER to continue:
```

Caso queiram alterar alguma informação, digitem o número da categoria +



<ENTER>. Para continuar o processo, simplesmente tecle <ENTER>.

```
-//-
*********************************************************************

 Done. The new package has been saved to

 /usr/local/src/Aplicativo-1.0.0/Aplicativo-1.0.0-i386-1.tgz
 You can install it in your system anytime using:

      installpkg Aplicativo-1.0.0-i386-1.tgz

*********************************************************************

# _
```

Finalmente, o programa programa já se encontra disponível e empacotado (*/usr/local/src/Aplicativo-1.0.0/Aplicativo-1.0.0-i386-1.tgz*), bastando apenas utilizar as ferramentas nativas do sistema para instalá-lo...

```
# installpkg Aplicativo-1.0.0-i386-1.tgz
```

... ou para removê-lo...

```
# removepkg Aplicativo-1.0.0-i386-1.tgz
```

RPM2TGZ

O utilitário *rpm2tgz* nada mais é que um conversor de pacotes do formato *RPM* para o formato nativo utilizado no *Slackware* (*.tgz*), que o acompanha.

A UTILIZAÇÃO

Sua utilização é bastante simples e prática. Veja sua sintaxe:

```
# rpm2tgz [NOME_DO_PACOTE]-[VERSÃO]-[ARQUITETURA].rpm
```

Acreditamos que não há necessidade de maiores instruções...

Segue um exemplo básico mostrando o processo de conversão:

```
# ls -l quake*
total 285
-r--r--r--    1 root     root       208483 Jul 19 14:57 quake-1.1-6cl.i386.rpm

root@darkstar:/# rpm2tgz quake-1.1-6cl.i386.rpm

root@darkstar:/# ls -l quake*
total 489
-r--r--r--    1 root     root       208483 Jul 19 14:57 quake-1.1-6cl.i386.rpm
-rw-r--r--    1 root     root       206511 Jul 19 15:02 quake-1.1-6cl.i386.tgz

# _
```

Agora é só eliminar a versão *.rpm* e instalar normalmente o pacote *.tgz*.

```
# installpkg quake-1.1-6cl.i386.tgz
Installing package quake-1.1-6cl.i386...
PACKAGE DESCRIPTION:

# _
```

Uma situação interessante ocorreu quando necessitamos remover este pacote apenas com o comando *removepkg* e o nome do pacote. Funcionava perfeitamente com outros programas, mas em algumas circunstâncias poderão ocorrer alguns problemas. Veja o seu processo desinstalação:

```
# removepkg quake

No such package: /var/log/packages/quake. Can't remove.
```

Para desinstalá-lo, foi necessário a utilização do nome completo e versão do pacote convertido, ao passo que os pacotes nativos desenvolvidos para o *Slackware* é necessário somente a utilização do nome do pacote:

```
# removepkg quake-1.1-6cl

Removing package /var/log/packages/quake-1.1-6cl.i386...
Removing files:
  --> Deleting /etc/sysconfig/quake
  --> Deleting /usr/bin/quake
  --> Deleting /usr/bin/squake
  --> Deleting /usr/doc/quake-1.1/readme.squake
  --> Deleting /usr/lib/quake/id1/config.cfg
  --> Deleting /usr/lib/quake/id1/netgame.cfg
  --> Deleting empty directory /usr/lib/quake/id1/
  --> Deleting empty directory /usr/lib/quake/
  --> Deleting empty directory /usr/doc/quake-1.1/
  --> Deleting empty directory /etc/sysconfig/
# _
```

As limitações

Infelizmente o *rpm2tgz* possui alguns inconvenientes de compatibilidade durante a conversão. Por exemplo, não conseguimos rodar a *API Wine* após converter seu pacote *.rpm* para o formato nativo do *Slackware*. Em sua execução, o programa acusou a falta de algumas bibliotecas do sistema, porém ao instalar o mesmo aplicativo no formato *RPM*, o programa executou normalmente.

Alien

- ✔ <http://kitenet.net/programs/alien/>.

Outro excelente conversor de pacotes que, diferente do *rpm2tgz*, converte de e para diversos formatos existentes.

A instalação

No diretório-raíz do programa, digite...

```
# perl Makefile.PL
Writing Makefile for Alien
# _
```

... onde em poucos instantes será gerado o arquivo *Makefile*. Para concluir a compilação e realizar a instalação...

```
# make && make install (ou checkinstall -y)
```

A UTILIZAÇÃO

A utilização do *Alien* é simples e prática, onde bastará apenas definir o arquivo origem e com um simples parâmetro, definir em qual formato deverá ser convertido. Segue sua sintaxe básica:

```
# alien [OPÇÕES] [ARQUIVO.FORMATO]
```

Onde:

Alien	
-d (--to-deb)	Converte para o formato nativo do *Debian*.
-r (--to-rpm)	Converte para o formato *Red Hat Package Management*.
-t (--to-tgz)	Converte para o formato nativo do *Slackware*.

Segue um simples e prático exemplo de conversão. Como utilizamos a distribuição *Slackware*, daremos preferência à conversão neste formato.

Para converter um pacote no formato *RPM* para o *Slackware*:

```
# alien -t [PACOTE].rpm
```

Já do formato *Debian*:

```
# alien -t [PACOTE].deb
```

Sem maiores complicações, basta utilizar as ferramentas nativas para instalar o novo pacote gerado.

```
# installpkg [PACOTE].tgz
```

RECOMENDAÇÕES

Procurem realizar a conversão de outros formatos de pacotes em última instância, pois infelizmente a maioria dos programas compilados no formato original foram concebidos para atenderem à uma determinada distribuição (ou conjunto delas). Dentre os principais problemas que poderemos ter estão na obtenção de um código compilado em outra versão do *GCC*, além da localização diferenciada das pendências necessárias ou ainda que necessitam de uma versão *glibc* diferente[2]. Se isto já ocorre com os programas compilados no formato nativo de outras versões do *Slackware*, imagine entre distribuições diferentes...

Já a conversão de pacotes compilados e convertidos para o formato *.tgz* pelo *Checkinstall* não só deixa de gerar problemas, visto que as ocorrências

2 Temos exemplos práticos, como alguns *plugins* não funcionam com navegadores específicos, enquanto que o aplicativo *X* pode requerer uma biblioteca que na distribuição *Y*, diferente do *Slackware*, se encontra em um diretório *Z*, ou ainda o programa *K* exige a versão *Q* da *glibc*, que também não é compatível com a versão da distribuição usada.

acima citadas não se enquadram nesta categoria, como também recomendamos a sua utilização, pois além de padronizar o gerenciamento geral dos programas, teremos em mãos todos os recursos disponíveis das ferramentas do *Slackware* para realizar quaisquer intervenções necessárias. Além disso, contaremos com um ótimo recurso, que é o da utilização do sumário para a inclusão de detalhes específicos.

Conclusão

Por *"tradição"*, a instalação de programas no *Slackware* é feita diretamente à partir do código-fonte do aplicativo desejado, quando estes não estavam disponíveis no *FTP* oficial da distribuição ou na página eletrônica da *LinuxPackages*. Por este motivo, são poucos os usuários que preferem realizar a conversão de outros pacotes para o formato nativo, muitas vezes esta atividade é feita somente em última instância. Sendo bons *slackers*, evitem utilizar os processos de conversão, ou somente utilizem-no quando houver realmente necessidade! &;-D

VI. Obtendo pacotes para o Slackware

Introdução

Muitas vezes chega um momento que a necessidade da realização de uma atividade requer a instalação de seu respectivo pacote. E a primeira pergunta que se faz é: onde irei obter o pacote desejado?

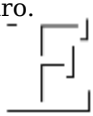
O FTP do Slackware

O *FTP* do *Slackware* estão os principais programas utilizados pela distribuição, além de diversos outros pacotes.

Os diretórios que compõe a estrutura de pacotes da distribuição são:

O FTP Slackware	
bootdisk/	As imagens de vários *kernels* para a criação de disquetes de inicialização, cada uma com sua finalidade e suporte específico.
extra/	Este diretório contém todos os aplicativos que não foram inclusos no *CD-ROM* de instalação por mera questão de falta de espaço.
isolinux/	Todas as imagens necessárias para a inicialização do sistema à partir do *CD-ROM*.
kernels/	Contém uma série de *kernels* compilados para diversas finalidades. O mais utilizado é o *bare.i*, que provê suporte aos dispositivos mais comuns.
pasture/	Pacotes de antigas versões do *Slackware* que não constam na atual. Dependendo da necessidade do usuário, muitos destes podem ser úteis.
rootdisk/	As imagens do *rootdisk* necessárias para a inicialização do sistema por disquetes.
slackware/	Este é o principal diretório do *FTP* do *Slackware*.
source/	Código-fonte das aplicações contidas nesta árvore. Utilizado especialmente por desenvolvedores e usuários que desejam obter ajustes mais finos do sistema com a compilação destes.
zipslack/	Contém o *ZipSlack*, uma versão compacta do sistema para a sua execução a partir de um *ZIP-drive*.

Na página oficial do *Slackware*, há uma seção informando os principais *FTPs* disponíveis pelo mundo inteiro.



Conteúdo da pasta principal Slackware

- ✔ <ftp://ftp.slackware.com/pub/slackware/>.

Os pacotes presentes na pasta principal do *Slackware* são os pacotes oficiais da distribuição. Eles também se encontram disponíveis nos *CD-ROMs* de instalação e extra.

Pasta Principal	
a/	Todos os pacotes essenciais para o funcionamento do *Slackware*, porém praticamente sem aplicativos, utilitários e interfaces gráficas, necessitando da instalação dos demais.
ap/	Todas as aplicações em modo texto. Porém para garantir o funcionamento do sistema, existe um editor de textos que consta no diretório */a* ao invés de estar aqui.
d/	Compiladores e bibliotecas necessários para a instalação de aplicativos à partir do código-fonte.
e/	Editor *Emacs*.
f/	Conjunto de *FAQs* – Perguntas mais freqüentes (em inglês) – para o *Slackware*, além de outras documentações importantes.
k/	Código-fonte do *kernel*.
kde/	Pacotes que compõe o *KDE* e demais aplicativos baseados na biblioteca *Qt*. Estes pacotes também se encontram no *CD-ROM Extra* do *Slackware*.
kdei/	Pacotes de internacionalização do *KDE* (*CD-ROM Extra*).
l/	Bibliotecas extras. Necessárias para vários programas.
n/	Pacotes para trabalhar com redes e seus aplicativos.
t/	Processador de textos *TEX*.
tcl/	Pacotes *TCL/tk*.
x/	Conjunto de pacotes que compõem o servidor *XFree86*, incluindo as fontes, a documentação e o código-fonte.
xap/	Aplicativos gerais para a interface gráfica, além de diversos outros ambientes gráficos.
y/	Jogos *BSD* e derivados.

A pasta extra/

Apesar dos principais programas esterem inclusos nos *CD-ROMs* de instalação, muitos destes, por questão de espaço e necessidade, não se encontram presentes. Foi criado então no *2o. CD-ROM* um diretório

chamado *extra/*, que contém programas adicionais que geralmente são muito úteis para o desenvolvimento das atividades no sistema.

À partir da versão *9.1*, o *Slackware* passou a incluir o *KDE* e o *GNOME* no *2o. CD-ROM Extra* da distribuição, onde o mesmo é solicitado para instalar estes ambientes gráficos durante a instalação do sistema.

A árvore corrente (atual)

- ✔ <*ftp://ftp.slackware.com/pub/slackware/slackware-current/*>.

Todos os pacotes que irão incorporar a nova distribuição do *Slackware* se encontram na árvore *slackware-current* de seu *FTP*. Para aqueles que desejam obter uma versão mais atualizada dos programas que compõe à distribuição, este é um local ideal para dar seu pontapé.

Correções de segurança e defeitos

- ✔ <*ftp://ftp.slackware.com/pub/slackware/slackware-11.0/patches/package/*>.

Por último, para a atualização de erros e diversas outras correções necessárias, existem as atualizações que também podem ser obtidos diretamente no *FTP* oficial do *Slackware*. Basta apenas baixar o pacote desejado e utilizar o comando *upgradepkg* para a atualização.

Outras fontes para a obtenção de pacotes

Dentre outras excelentes fontes para a obtenção de pacotes, destacam-se:

Repositórios e espelhos Slackware

- ✔ <*http://www.slackware.com/getslack/*>.
- ✔ <*http://www.linuxpackages.net/*>.

Em formato .RPM

Para obtermos pacotes no formato *.RPM*, existem as páginas...

- ✔ <*http://www.rpmfind.net/*>.
- ✔ <*http://www.rpmseek.net/*>.

Hospedagem de projetos

Já a *SourceForge.net* e a *FreashMeat* são páginas eletrônicas ne renome internacional especializadas em hospedagem de projetos livres.

- ✔ <*http://www.freashmeat.net/*>.
- ✔ <*http://sourceforge.net/*>.

Código Livre

A *CódigoLivre* é uma iniciativa da *Univates* onde o principal objetivo é...

> *"fomentar o uso do software livre no Brasil, auxiliando os desenvolvedores, oferecendo-lhes todas as ferramentas necessárias ao bom andamento de um projeto, ao mesmo tempo em que oferece à comunidade usuária um portal onde podem obter softwares de qualidade." -> ["Quem Somos", por Código Livre].*

- ✔ *<http://codigolivre.org.br/>.*

TcheLinux

No *Brasil*, temos na página da *TcheLinux* matérias interessantes, além de disponibilizar um atalho para diversos pacotes para o *Slackware*.

- ✔ *<http://slackpacks.tchelinux.com.br/>.*

Slackware-Brasil

Também existe um *FTP* brasileiro, o *Slackware Brasil*, que apenas mantém os pacotes que compõe a distribuição oficial.

- ✔ *<ftp://ftp.slackwarebrasil.org/pub/linux/slackware/>.*

SlackLife

O grupo *SlackLife* também lançou recentemente um espelho onde disponibiliza os pacotes do *Slackware* e imagens *ISO* das distribuições *live-CDs* como o *Slax* e *Goblinx*.

- ✔ *<http://mirror.slacklife.com.br/>.*

SuperDownloads

Dentre outras páginas interessantes, também se encontra a *SuperDownload*, que apesar de não focar exclusivamente os sistemas *GNU/Linux*, dispõe de uma seção reservada para os *linuxers*. &;-D

- ✔ *<http://superdownloads.ubbi.com.br/linux/>.*

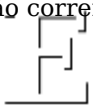
Tux Resources

Esta última conhecemos recentemente. A *Tux Resources* possui uma seção para *download* bem organizada aos mesmos moldes da *SuperDownload*. Lá encontraremos os programas desejados, todos classificados por categorias.

- ✔ *<http://www.tuxresources.org/>.*

-//-

Caso ainda possuam dificuldade em encontrar os pacotes desejados, deveremos ir à página principal do desenvolvedor e procurar obter pelo menos o código-fonte dos programas desejados – isto se não houver um pacote pré-compilado para a versão corrente do *Slackware*.



A INTEGRIDADE DOS PACOTES BAIXADOS

Dependendo da conexão utilizada, muitas vezes o conteúdo dos arquivos e pacotes baixados da *Intenet* não se encontram íntegros, face às interferências da linha telefônica durante a realização do processo, entre outro possíveis problemas. Para checar a integridade destes arquivos, utilizem o *md5sum*.

MD5SUM

O *md5sum* é um aplicação desenvolvida para checar a integridade física de qualquer conteúdo trafegado na *Internet*; qualquer alteração na consulta destes, por mínima que seja, logo é apontado pelo aplicativo.

Observe que no local onde os pacotes encontram-se disponíveis, existes arquivos-textos com a extensão *.md5*, o qual em seu respectivo conteúdo encontra-se uma seqüência de *32* dígitos. Estas seqüências são utilizadas pelo comando *md5sum* para realizar a checagem de integridade dos arquivos baixados. Sua sintaxe é a seguinte:

```
$ md5sum [ARQUIVO_BAIXADO]
```

Basta apenas verificar a cadeia de caracteres exibida e comparar com a chave que se encontra disponível junto do arquivo baixado, geralmente na página do distribuidor.

```
$ md5sum firefox-0.8-i686-linux-gtk2+xft-ptBR.tar.gz
8cabf90c4f6c353d2c9bca758ef813bc  firefox-0.8-i686-linux-gtk2+xft-
ptBR.tar.gz
$ _
```

Observe que o aplicativo retornou uma seqüência de caracteres, o qual deverá ser conferida com a chave disponibilizada. Se a seqüência estiver exatamente igual, isto significa que seu pacote se encontra íntegro, podendo ser utilizado normalmente. Caso contrário, ocorreu algum erro do qual resultou no corrompimento do pacote baixado, e neste caso será necessário baixá-lo novamente.

NOMENCLATURA DOS PACOTES

A nomenclatura dos arquivos empacotados obedecem a um formato especificado, tendo separadas suas definições apenas por hífen e ponto:

```
Nome-Versão-Correção (Release).Plataforma.Formato (extensão)
```

Utilizaremos como exemplo o pacote fictício *Darkstar-1.0-1.i386.rpm*:

Nomenclatura de pacotes	
Nome	*Darkstar*.
Versão	*1.0*.
Correção	*1*.

Nomenclatura de pacotes	
Plataforma	*i386*.
Formato	Indicado pela extensão: – pacote padrão *Red Hat* (*.rpm*). Poderá também ser *Slackware* (*.tgz*) ou *Debian* (*.deb*).

Em muitos casos, dada a compilação específica de um determinado aplicativo para uma distribuição baseada na *Red Hat,* são adicionados *2* caracteres para informar a distribuição do qual este pacote é compatível.

- *cl -> Conectiva Linux*;
- *mk -> Mandrake Linux*;

Para exemplo:

- *quake-1.1-6cl.i386.rpm*;
- *qt-2.2mk.i586.rpm*;

Existem outras nomenclaturas que poderão existir. Vejam alguns exemplos:

Darkstar-0.6beta-i686.rpm

Trata-se de um pacote que se encontra na versão *0.6*, porém em fase de testes (*beta*), compilado na arquitetura *i686*, que corresponde à *2a.* geração do *Pentium* e empacotado no formato *RPM*. Para máquinas de produção (que necessitam de programas estáveis), geralmente não é recomendável a instalação destes pacotes.

Darkstar-1.0-noarch.tgz

Pacote que se encontra na versão *1.0*, sem correção e que é compatível com qualquer arquitetura existentes (*noarch*), empacotado no formato nativo do *Slackware*. Somente pacotes que contém o código-fonte ou arquivos textos das aplicações é que pertencem à esta categoria de arquitetura universal. O pacote de internacionalização do *KDE* e os fonteS do *kernel* são exemplos.

Conclusão

Um dos maiores inconvenientes das distribuições que possuem inúmeros pacotes está na desinstalação dos mesmos, pois a maioria dos usuários não utilizam e sequer sabem se poderão ser removidos ou não. Apesar do *Slackware* não incluir em sua distribuição uma grande variedade de pacotes, boa parte das necessidades dos usuários são satisfeitas com os que se encontram disponíveis nela. Estes deverão apenas procurar as aplicações restantes para complementar o sistema, mesmo que isto implique em uma maior incidência de procura dos demais programas. Em geral, isto será benéfico para a otimização geral do sistema, onde apenas iremos manter apenas o que é essencial. &;-D

ENCERRAMENTO

Infelizmente, o gerenciamento de programas nos sistemas *GNU/Linux* não é um processo tão simples e de fácil compreensão. Existem diversos aspectos e necessidades para garantir uma boa instalação e o perfeito funcionamento do programa que se deseja instalar. Porém, para a nossa alegria, uma vez tendo estes procedimentos realizados corretamente, o sistema em geral tende à fornecer uma ótima performance e estabilidade, em comparação à outros sistemas operacionais...

Apesar da necessidade de conhecimento inerentes aos diferentes processos de instalação, a principal vantagem obtida é a possibilidade de ter um controle total sobre todos os aplicativos que encontram-se instalados no sistema. A eficiência e facilidade da atualização e até mesmo da remoção de pacotes antigos nos possibilita condicionar o sistema para os nossos propósitos sem maiores riscos de conflitos e erros, desde que estas operações sejam realizadas com conhecimento e consciência. Nada de uma biblioteca mal comportada, conflitante ou não removida por causa de uma desinstalação mal feita, como ocorre em muitos sistemas operacionais. São poucos os casos de sistemas *GNU/Linux* "*saturados*" com a instalação e desinstalação de aplicativos, onde a ocorrência de instabilidade e travamentos obrigam os usuário à reinstalarem novamente o sistema para eliminarem tais problemas.

Dentre todos os processos existentes, a compilação direta do código-fonte está entre as mais complexas e ao mesmo tempo a que fornece excelentes resultados quando bem empregados. Por este motivo, recomendamos aos mais experimentados utilizarem este processo quando houver a ausência de um pacote pré-compilado oficial e compatível com a distribuição. Sabendo-se das possibilidades de aprendizado proporcionadas pelo processo de compilação – e possibilidade da falta de aplicações pré-compiladas –, as instruções de instalação da grande maioria dos programas – em especial contidas na parte *Aplicativos & utilitários* – foram desenvolvidas enfocando principalmente este processo.

Antes, iremos conhecer agora os principais ambientes gráficos disponíveis para os sistemas *GNU/Linux* (é isso mesmo, são vários e não um único!), de forma que possamos habituarmos com seus recursos e aí sim, partir para um dos maiores desafios na administração deste maravilhoso sistema operacional: seleção, instalação e administração de aplicativos! &;-D