

5A. PARTE!

GERENCIAMENTO DE PROGRAMAS

✓ *Copyright (c) 2002-2008 -Ednei Pacheco de Melo.*

Permission is granted to copy, distribute and/or modify this document under the terms of the *GNU Free Documentation License*, version 1.1 or any later version published by the *Free Software Foundation*; a copy of the license is included in the section entitled "*GNU Free Documentation License*".

ÍNDICE

VISÃO GERAL.....	5
I. FERRAMENTAS DE GERENCIAMENTO.....	6
Introdução.....	6
As ferramentas.....	6
Slackware Package Tools.....	6
Current.....	7
Other.....	7
Floppy.....	7
Remove.....	7
View.....	8
Setup.....	9
Na linha de comando.....	9
installpkg.....	10
removepkg.....	10
updatepkg.....	11
Recomendações gerais.....	11
Conclusão.....	12
II. FERRAMENTAS PARA A ATUALIZAÇÃO.....	13
Introdução.....	13
O Slackpkg.....	13
A instalação.....	13
A configuração.....	13
/etc/slackpkg/slackpkg.conf.....	14
/etc/slackpkg/mirrors.....	17
/etc/slackpkg/blacklist.....	18
A utilização.....	18
Observações.....	21
Recomendações gerais.....	21
Conclusão.....	22
III. A COMPILAÇÃO DO CÓDIGO-FONTE.....	23
Introdução.....	23
Considerações básicas.....	23
O que é o processo de compilação?.....	23
A compilação estática e a dinâmica.....	23
As ferramentas do Projeto GNU.....	24
As bibliotecas.....	24
GNU C Library.....	24
Libtool.....	25
As ferramentas de automação.....	25
m4.....	25
automake.....	25
autoconf.....	26
make.....	26

Os compiladores.....	26
<i>GNU C Compiler</i>	26
Entre outras.....	27
<i>binutils</i>	27
<i>patch</i>	27
A compilação padrão.....	27
Obtendo o código-fonte.....	28
Descompactando o código-fonte.....	28
O local de armazenamento.....	28
A leitura de documentações.....	29
A configuração, a compilação e a instalação.....	29
Limpando o diretório do pacote compilado.....	30
Desinstalando um pacote compilado.....	30
Considerações avançadas.....	30
Conteúdo da documentação.....	31
<i>O diretório docs</i>	31
<i>Readme / Install</i>	31
<i>Copying</i>	33
<i>Copyright</i>	34
<i>Release</i>	34
<i>Credits</i>	35
<i>Changelog</i>	35
Funcionalidades detalhadas.....	35
<i>./configure</i>	36
<i>make</i>	37
<i>make deps / make depend</i>	37
<i>make install</i>	37
<i>make clean</i>	37
<i>make uninstall</i>	37
Observações importantes.....	38
<i>Manutenção do código-fonte</i>	38
<i>Aplicativos & utilitários</i>	38
<i>Drivers, módulos e kernel</i>	38
Conclusão.....	39
IV. A CONVERSÃO DE PACOTES E PROGRAMAS.....	40
Introdução.....	40
As ferramentas.....	40
Alien.....	40
CheckInstall.....	41
rpm2tgz.....	44
Recomendações.....	45
Conclusão.....	45
V. OBTENDO OS PACOTES OFICIAIS.....	46
Introdução.....	46
O FTP do Slackware.....	46
bootdisks.....	46
extra.....	46

isolinux.....	46
kernels.....	47
pasture.....	47
patches.....	47
rootdisk.....	47
slackware.....	47
source.....	48
testing.....	48
zipslack.....	49
Sobre a árvore /current.....	49
Checando a integridade dos pacotes.....	49
md5sum.....	49
Conclusão.....	50

VISÃO GERAL

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 binários compilados (ou não) de um programa, arquivado em um formato especial reconhecido pelo gerenciador de pacotes da distribuição. 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. Tais pacotes são chamados de pendências.

As pendências possuem suas particularidades de acordo com a aplicação; 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 a 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.

Embora o gerenciamento de pacotes em sistemas *GNU/Linux* requeira certo conhecimento técnico, existem vantagens interessantes a serem consideradas: maior eficiência na administração de todo o processo (instalação, atualização, verificação e remoção), com poucas probabilidades de conflitos e erros, além da flexibilidade em termos de customização do conjunto de pacotes necessários. Por isso, são poucos os casos de sistemas *GNU/Linux* "saturados" com a instalação e/ou desinstalação de aplicativos, causando instabilidades e travamentos e assim, obrigando os usuário a reinstalarem novamente o sistema.

Nos próximos capítulos, conheceremos a fundo todo o processo de gerenciamento de programas no *Slackware*. &;-D



I. FERRAMENTAS DE GERENCIAMENTO

INTRODUÇÃO

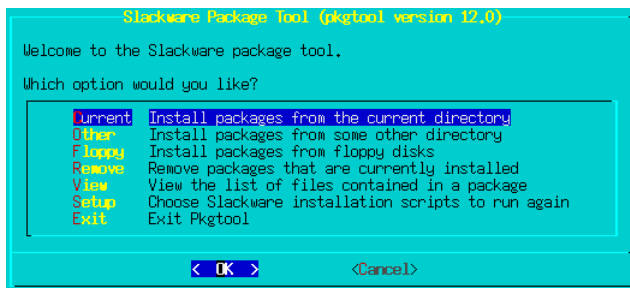
Não muito diferente das demais distribuições, o *Slackware* possui também seu próprio gerenciador de pacotes pré-compilados, além de diversas ferramentas - apesar de não terem algumas funcionalidades importantes, como a checagem de dependências e a possibilidade de realizar a atualização de todo o sistema através de uma conexão em rede ou *Internet*.

Neste capítulo, iremos conhecer tais ferramentas, onde entra em destaque, o *Slackware Package Tools*, que é conhecido popularmente como *Pkgtool*.

AS FERRAMENTAS

SLACKWARE PACKAGE TOOLS

O *Slackware Package Tools* - o chamaremos aqui como *Ppkgtool* - é o gerenciador de pacotes padrão do *Slackware*, escrito basicamente com a utilização de *scripts* (arquivos de lote) e funciona somente em modo texto.



Interface principal do Slackware Package Tool.

Além das funcionalidades básicas essenciais, o *Pkgtool* também provê uma série de *scripts* de configuração que auxiliam muito para a realização de ajustes e manutenções gerais do sistema.

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

```
# pkgtool
```

O *Pkgtool* provê 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*, *Other*, *Floppy*, *Remove*, *View* e *Setup*.

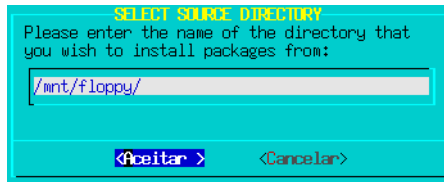


CURRENT

A opção *Current* instala os pacotes pertencentes ao diretório onde este utilitário é invocado. Por exemplo, se entrarmos em nosso diretório `/mnt/pkg/slack` e a utilizarmos, todos os pacotes nativos 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.

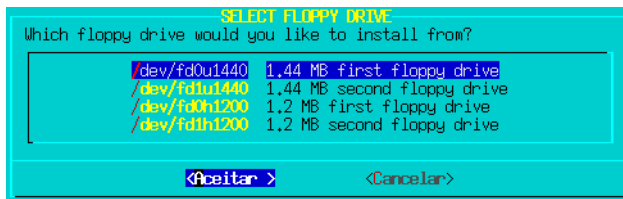
OTHER

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



FLOPPY

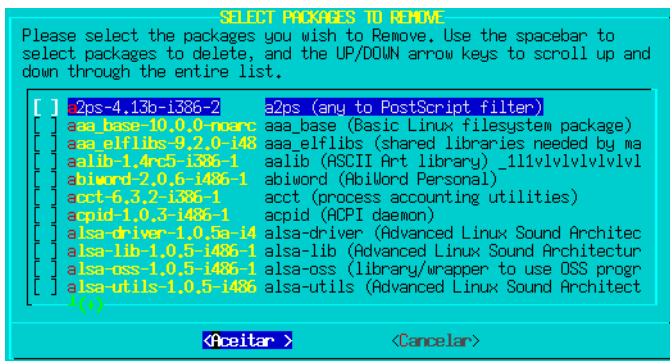
A opção *Floppy* realiza a instalação dos pacotes desejados, porém estes tendo como origem a unidade de disquetes. Basta selecionar a unidade que possui o disquete com os pacotes desejados.



REMOVE

A opção *Remove* - como o próprio nome diz - remove os pacotes desejados conforme uma seleção pré-realizada.

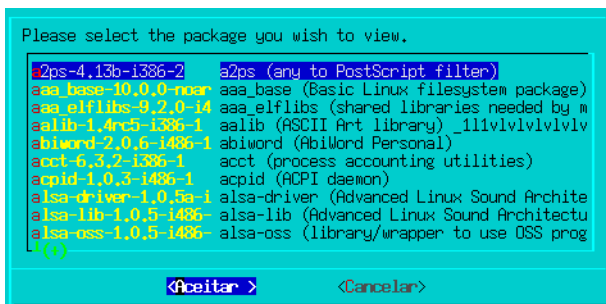




Acionando a <BARRA_DE ESPAÇO>, marcaremos e/ou desmarcaremos os pacotes que desejamos desinstalar e/ou manter.

VIEW

A opção *View* exibe as informações referentes ao pacote selecionado, onde antes, será mostrado uma listagem de pacotes previamente instalados.



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



```

CONTENTS OF PACKAGE: pkgtools-10.0.0-i486-1
PACKAGE NAME:      pkgtools-10.0.0-i486-1
COMPRESSED PACKAGE SIZE: 179 K
UNCOMPRESSED PACKAGE SIZE: 950 K
PACKAGE LOCATION: /var/log/mount/slackware/a/pkgtools-10.0.0-i486-1.tgz
PACKAGE DESCRIPTION:
pkgtools: pkgtools (The Slackware package maintenance system)
pkgtools:
pkgtools: This package contains utilities for handling Slackware packages.
pkgtools: Included are the command line utilities 'installpkg', 'removepkg',
pkgtools: 'makepkg', 'explodepkg', and 'upgradepkg' that install, remove,
pkgtools: build, examine, and upgrade software packages. Also included are
pkgtools: 'pkgtool', a menu based program for installing packages, removing
pkgtools: packages, or viewing the packages that are installed on the system
pkgtools: documentation (man pages), and a few other system admin scripts.
pkgtools:
pkgtools:
FILE LIST:
./
bin/
bin/dialog

```

(6%)

< Sair >

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

SETUP

A opção *Setup* disponibiliza uma série de *scripts* para a configuração, tal como é feito no processo de instalação da distribuição.

```

SELECT SYSTEM SETUP SCRIPTS
Please use the spacebar to select the setup scripts to run. Hit
enter when you are done selecting to run the scripts.

```

04.mkfontdir	Run mkfontdir and mkfontscale in font dire
05.fontconfig	Run fc-cache to locate new fonts for Xft
07.update-desktop	Run update-desktop-database.
07.update-mime-da	Run update-mime-database.
70.install-kernel	Install a Linux kernel from a bootdisk
80.make-bootdisk	Create a USB Linux boot stick
90.modem-device	Select modem device
htalview	Set a default browser link.
liloconfig	Set up LILO to boot Linux (and other OSese
v(i)	

< OK > <Cancel>

Deveremos marcá-los com a tecla <BARRA_DE_ESPAÇO> e teclar <ENTER> em seguida para que eles sejam executadas sequencialmente.

Poderemos obter mais informações adicionais sobre a utilização destas sub-opções durante a consulta dos capítulos referente aos processos de ajustes e configurações do sistema. Estas informações situam-se na *3a. Parte: A Instalação* e *4a. Parte: Ajustes & Configurações*.

NA 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*. Para esta literatura, descreveremos apenas os três primeiros comandos.



INSTALLPKG

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

Sintaxe:

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

Exemplo:

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

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

REMOVEPKG

Tal como o comando *installpkg*, utilizamos o comando *removepkg*, que por sua vez, tem a função de remover o pacote desejado.

Sintaxe:

```
# removepkg [PACOTE]
```

Exemplo:

```
# removepkg libdvddread
```

```
Removing package /var/log/packages/libdvddread-0.9.4-i686-1...
Removing files:
--> Deleting symlink /usr/lib/libdvddread.so
--> Deleting symlink /usr/lib/libdvddread.so.3
--> Deleting /usr/doc/libdvddread-0.9.4/AUTHORS
--> Deleting /usr/doc/libdvddread-0.9.4/ChangeLog
--> Deleting /usr/doc/libdvddread-0.9.4/COPYING
--> Deleting /usr/doc/libdvddread-0.9.4/INSTALL
--> Deleting /usr/doc/libdvddread-0.9.4/NEWS
--> Deleting /usr/doc/libdvddread-0.9.4/README
--> Deleting /usr/doc/libdvddread-0.9.4/TODO
--> Deleting /usr/include/dvddread/dvd_reader.h
--> Deleting /usr/include/dvddread/ifo_print.h
--> Deleting /usr/include/dvddread/ifo_read.h
```

```
--> Deleting /usr/include/dvdrread/ifo_types.h
--> Deleting /usr/include/dvdrread/nav_print.h
--> Deleting /usr/include/dvdrread/nav_read.h
--> Deleting /usr/include/dvdrread/nav_types.h
--> Deleting /usr/lib/libdvdrread.a
--> Deleting /usr/lib/libdvdrread.la
--> Deleting /usr/lib/libdvdrread.so.3.0.0
--> Deleting empty directory /usr/include/dvdrread/
--> Deleting empty directory /usr/doc/libdvdrread-0.9.4/
```

```
# _
```

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. Notem ainda que, em `/var/log/packages/` são mantidos arquivos-textos com os nomes dos pacotes que foram excluídos, além de todas as informações gerais provenientes de tais pacotes.

UPDATEPKG

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 e especificados através do curinga '*' <ASTERÍSCO>). Mas não é o ideal para atualizar todos os pacotes do sistema. Para esta atividade, temos ótimas ferramentas, como o *Slackpkg*.

RECOMENDAÇÕES GERAIS

Uma das características marcantes do *Slackware* está no fato de que seu gerenciador de pacotes padrão não possui 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/`, 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 processos. 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.

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

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, poderão ocorrer 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 para a desenvoltura destes processos, pois nestas ocasiões, serão eles a base principal para a solução da maioria das dificuldades. &:-D



II. FERRAMENTAS PARA A 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 temos ótimas ferramentas desenvolvidas para esta necessidade. Nesta literatura, destacaremos o *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 automatizar a atualização dos pacotes oficiais do *Slackware*. Basicamente as atividades deste programa consiste em checar o sistema, verificar quais os pacotes se encontram instalados e baixar suas respectivas atualizações. Instalações e remoções também poderão ser feitas.

A INSTALAÇÃO

Para obtermos o *Slackpkg*, deveremos baixá-lo da página oficial do projeto, mais especificamente na série de pacotes */extra*; ele também pode ser obtido nas mídias de instalação da distribuição. A *1a.* opção será a mais recomendada por estar atualizada, especialmente os *CD-ROMs* estiverem com alguns meses de idade...

A ferramenta se encontra empacotada no formato nativo da distribuição; portanto, utilizem o procedimento padrão de instalação do *Slackware*:

```
# installpkg slackpkg-[VERSÃO]-noarch-[REV].tgz
```

A CONFIGURAÇÃO

Todas as opções de ajuste e configuração da ferramenta estão gravadas em três arquivos de configuração, que por sua vez se encontram armazenados no diretório */etc/slackpkg/*:

```
$ cd /etc/slackpkg
$ ls -l
total 40
-rw-r--r-- 1 root root 798 2007-03-12 22:29 blacklist
-rw-r--r-- 1 root root 29838 2007-03-12 22:29 mirrors
-rw-r--r-- 1 root root 3565 2007-03-12 22:29 slackpkg.conf
$ _
```



Estes três arquivos são o *blacklist*, o *mirrors* e o *slackpkg.conf*.

/ETC/SLACKPKG/SLACKPKG.CONF

O *slackpkg.conf* mantém as definições gerais de ajuste e configuração.

```
# SlackPkg - An Automated packaging tool for Slackware Linux
# Copyright (C) 2003,2004,2005,2006,2007 Roberto F. Batista, Evaldo Gardenali
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
#
# Project Page: http://slackpkg.org/
# Roberto F. Batista (aka PiterPunk) piterpk@terra.com.br
# Evaldo Gardenali (aka UdontKnow) evaldogardenali@fasternet.com.br
...
```

De início, ele apresenta uma observação pela qual as definições são mantidas em dois estados: habilitado (*on*) e desabilitado (*off*):

```
# For configuration options that have only two states, possible values are
# either "on" or "off"
```

Em virtude de sua grande popularidade, o *Slackware* é uma distribuição portada para diferentes arquiteturas. Por isso, se quisermos utilizar o *Slackpkg* em alguma outra plataforma em que o *Slackware* foi portado, teremos de ajustar estas definições na seguinte seção:

```
# Remember, the only official Slackware ports are x86 and s390, and
# slackpkg developers don't have s390 boxes for testing. If you are
# testing/using in other archs and have suggestions or patches,
# please let me know (piterpk@terra.com.br)
#
# Select the architecture of your system
# x86 (the main arch for Slackware)
ARCH=i[3456]86
# x86_64 (Slamd64 and BlueWhite64)
#ARCH=x86_64
# PowerPC (Slackintosh)
#ARCH=powerpc
# S/390 (Slack/390)
#ARCH=s390
```

Por padrão, o *Slackpkg* é definido para utilizar a arquitetura *x86*.

```
# x86 (the main arch for Slackware)
```

```
ARCH=i[3456]86
```

Se quisermos ajustar as definições para utilizar outro porte/arquitetura, bastará apenas comentarmos a definição *x86* e desmarcar aquela utilizada.

A seguir, vem a definição da localização onde os pacotes serão baixados:

```
# Downloaded files will be in directory below:  
TEMP=/var/cache/packages
```

Se quisermos utilizar outro diretório, bastará apenas redefinirmos o caminho utilizado, como por exemplo, */mnt/pkg/Slack*.

Já os arquivos que contém as definições de listagem dos pacotes são mantidos em */var/lib/slackpkg*:

```
# Package lists, file lists and others will be at WORKDIR:  
WORKDIR=/var/lib/slackpkg
```

Embora não vejamos utilidade em modificá-los, poderemos redefini-los da mesma forma em que fizemos no armazenamento de pacotes.

Para obter arquivos de repositórios *FTP*, o *Slackpkg* utiliza o *wget*. Logo, poderemos realizar aqui as definições de parâmetros para o uso geral:

```
# Special options for wget (default is WGETFLAGS="--passive-ftp")  
WGETFLAGS="--passive-ftp"
```

No caso acima, o *wget* está configurado para fazer a transferência no modo passivo. Poderemos tanto excluir esta definição, quanto acrescentar outras.

Todos os pacotes obtidos poderão ser mantidos ou não, conforme as nossas preferências. Por exemplo, para aqueles que fazem múltiplas instalações da distribuição, poderemos utilizar os pacotes já baixados para serem utilizados em outras máquinas. Nestas circunstâncias...

```
# If DELALL is "on", all downloaded files will be removed after install.  
DELALL=on
```

... definam a variável *DELALL* para o valor *off*, que por sua vez desabilita a remoção dos arquivos após a instalação.

Existe a possibilidade dos pacotes obtidos serem corrompidos durante a sua obtenção. E claro, pacotes corrompidos não deverão ser instalados!

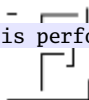
```
# If CHECKPKG is "on", the system will check the md5sums of all packages before  
# install/upgrade/reinstall is performed.  
CHECKPKG=on
```

Felizmente existe a variável *CHECKPKG*, que habilitada por padrão (valor *on*), evitará este tipo de ocorrência. Para isto, ela buscará as chaves geradas pelo *md5sums* para verificar a integridade dos pacotes.

O mesmo se dá no ato da verificação da procedência destes pacotes, ...

```
# If CHECKGPG is "on", the system will verify the GPG signature of each  
package
```

```
# before install/upgrade/reinstall is performed.
```



```
CHECKGPG=on
```

... onde a variável *CHECKGPG* também é mantida habilitada (valor *on*) para garantir que não venhamos a instalar pacotes de origens duvidosas.

Os pacotes que contém as correções (*patches*) são muito requisitados por aqueles que necessitam de manter o sistema atualizado em ambientes onde a segurança é prioritária:

```
# The lines below will set the download priority.
# Default values: /patches /slackware /extra /pasture /testing
FIRST=patches
SECOND=slackware
THIRD=extra
FOURTH=pasture
FIFTH=testing
```

Por isto, os pacotes a serem instalados com maior prioridade (*FIRST*) são aqueles que se encontram armazenados no diretório *patches*. Poderemos redefinir a ordem de prioridades, se assim desejarmos.

Em muitos espelhos, nem sempre a estrutura de diretórios dos pacotes e arquivos do *Slackware* são mantidos em um diretório principal chamado *slackware*. Isto acontece especialmente em distribuições derivadas:

```
# The default MAIN is "slackware", but some derived distros uses other
# names as the main directory. Of course, MAIN needs to be listed in
# the download priority directories
MAIN=slackware
```

Neste caso, será necessário redefinirmos a nomenclatura do diretório principal para o nome do diretório em uso do repositório escolhido, pois caso contrário, a ferramenta de atualização não irá funcionar a contento.

Ao atualizarmos os pacotes, poderemos apagar as definições feitas em seus arquivos de configurações. Para que isto não ocorra, o *Slackpkg* definiu a variável *POSTINST*, sendo esta mantida habilitada por padrão.

```
# Enables (on) or disables (off) slackpkg's post-installation features, such
# as checking for new (*.new) configuration files and new kernel images, and
# prompts you for what it should do. Default=on
POSTINST=on
```

Assim, a ferramenta irá perguntar se queremos manter as definições antigas, sobrescrever, renomear com a extensão *.new* ou apagá-las.

Embora seja uma ferramenta feita para ser utilizada em modo texto, a versão atual do *Slackpkg* utiliza caixas de diálogo para facilitar algumas operações, como a escolha dos pacotes a serem atualizados, por exemplo.

```
# The ONOFF variable sets the initial behavior of the dialog interface.
# If you set this to "on" then all packages will be selected by default.
# If you prefer the opposite option (all unchecked), then set this to "off".
ONOFF=on
```

Se mantermos esta definição habilitada, todos os pacotes serão marcados por padrão; se a desabilitarmos, teremos que selecioná-los.



Poderemos tanto baixar e atualizar todos pacotes “um-por-um” ou “todos-de-uma-vez” para serem instalados posteriormente:

```
# If this variable is set to "on", all files will be downloaded before the
# requested operation (install or upgrade) is performed.  If set to "off",
# then the files will be downloaded and the operation (install/upgrade)
# performed one by one.  Default=off
DOWNLOAD_ALL=off
```

Para quem usa conexão discada e nem sempre pode se manter conectado o tempo inteiro durante a obtenção dos pacotes, pode ser mais interessante habilitarmos a instalação dos pacotes apenas após a obtenção de todos eles.

Embora as caixas de diálogo em modo texto possam tornar mais intuitiva as operações, nem sempre agrada a todos.

```
# Enables (on) or disables (off) the dialog interface in slackpkg.  Default=on
DIALOG=on
```

Se for este o caso, poderemos desabilitá-la através da variável *DIALOG*.

Por último, falta definirmos qual o espelho que iremos utilizar.

```
# The MIRROR is set from /etc/slackpkg/mirrors
# You only need to uncomment the selected mirror.
# Uncomment one mirror only.
```

Para isto, existe o arquivo à parte chamado *mirrors*.

/ETC/SLACKPKG/MIRRORS

Após a instalação da ferramenta, será necessária a edição do arquivo de configuração */etc/slackpkg/mirrors*, onde deveremos definir qual o espelho que será utilizado para a obtenção das atualizações. Para isto, deveremos apenas desmarcar a listagem presente. Ou se desejarem, podem obter uma lista atualizada na página oficial do *Slackware*.

```
# You only need to select one mirror and uncomment them. Please,
# ONLY ONE mirror can be uncommented each time.
#
# In this file you can find mirrors for slackware 11.0, current
# and 10.2 (in this order).
#
# The mirrors file is kept synced with the official slackware
# mirrors. If you find any incorrect mirror, please report it
# to fizban <fizban@slackware.com>
```

Como podem ver, o *Slackpkg* utiliza somente um único espelho para obter os pacotes atualizados, apesar de suportar os protocolos *HTTP* e *FTP*. Portanto, desmarquem apenas um, senão o programa não irá funcionar.

```
# slackpkg update
```

```
Slackpkg only works with ONE mirror selected. Please, edit your
/etc/slackpkg/mirrors and comment one or more lines. Two or more
mirrors uncommented isn't valid syntax.
```



/ETC/SLACKPKG/BLACKLIST

Outro ajuste importante está no arquivo de configuração *blacklist*. Este por sua vez define quais os pacotes ou séries que não deverão ser atualizados.

```
# 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
#
# 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 can't be updated.
#
aaa_elflibs

#
# Now we can blacklist whole directories
# The two versions of udev inside that dir conflicts with slackware
# default udev (in /slackware).
/extra/udev-alternate-versions
```

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, haverá travamentos que impedirão a inicialização do sistema, nos obrigando à inicializar a máquina com os *CD-ROMs* de instalação para realizar a gravação do *LILO* na *MBR*.

A UTILIZAÇÃO

A sintaxe básica do programa para o gerenciamento básico de pacotes é:

```
# slackpkg [OPÇÕES] [PACOTE]
```

Onde:

Slackpkg	
<i>blacklist</i>	Para inserir um pacote na lista negra:
<i>download</i>	Para apenas baixar um pacote: <code># slackpkg download [PACOTE]</code>
<i>info</i>	Para exibir as informações de um pacote: <code># slackpkg info [PACOTE]</code>
<i>install</i>	Para instalar um pacote: <code># slackpkg install [PACOTE]</code>
<i>reinstall</i>	Para reinstalar um pacote: <code># slackpkg reinstall [PACOTE]</code>
<i>upgrade</i>	Para atualizar um pacote: <code># slackpkg upgrade [PACOTE]</code>
<i>remove</i>	Para remover um pacote: <code># slackpkg remove [PACOTE]</code>

Além dos parâmetros utilizados para a manipulação dos pacotes, o *Slackpkg* poderá realizar a atualização de todo o sistema através da execução da seguinte seqüência de comandos:

```
# slackpkg update
# slackpkg install-new
# slackpkg upgrade-all
# slackpkg clean-system
```

Para atualizar todo o sistema, deveremos digitar antes...

```
# slackpkg update
```

... para atualizar a lista de pacotes.

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

Logo em seguida, deveremos utilizar...

```
# slackpkg install-new
```

... para serem selecionados e instalados os novos pacotes que passaram a fazer parte da distribuição. Depois...

```
# slackpkg upgrade-all
```

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)keep 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:



- *K*: continuar com os arquivos antigos e novos, mantendo os novos como *.new* para posterior avaliação;
- *O*: sobrescrever os arquivos antigos pelos novos;
- *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 a serem obtidos e a taxa de transferência da conexão.

Por último, faz-se necessário rodar o comando...

```
# slackpkg clean-system
```

Será apresentada uma lista de pacotes que não fazem mais parte da distribuição, normalmente por causa da necessidade de removê-lo durante as atualizações do repositório. Tais pacotes devem ser marcados para serem removidos. Porém, poderão ser apresentados também pacotes instalados obtidos de outras fontes. Por exemplo, muitos necessitam da suíte de escritório *OpenOffice.org*; se esta estiver instalada com o uso de pacotes nativos, eles aparecerão na listagem. Neste caso, mantenham os pacotes desta suíte, se houver necessidade de utilizá-la.

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 que pertencentes a distribuição. Para os pacotes instalados de outras fontes ou compilados diretamente a 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 dos programas a serem atualizados. 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). Esta irá resguardar o arquivo original para que seja substituído por um novo arquivo com as novas definições.

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-las no novo. Se estivermos atentos a 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 torna-se inviável, dada as circunstâncias em que esta atividade é executada. Para isto, existem as ferramentas de atualização!

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



III. A COMPILAÇÃO DO CÓDIGO-FONTE

INTRODUÇÃO

A compilação do código-fonte disponível de um determinado programa ou *driver*, apesar de não ser tão fácil quanto a utilização de binários pré-compilados, é considerada como uma excelente oportunidade de garantir a boa implementação e a excelência de certos resultados. Graças a uma correta e eficiente definição de parâmetros, 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

O QUE É O PROCESSO DE COMPILAÇÃO?

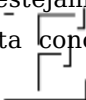
A compilação é o processo pelo qual as instruções contidas no código-fonte são convertidas para uma linguagem de máquina, que é somente entendida pelo próprio computador. Para os usuários mais novatos, vindos do sistema operacional *Windows*, muitos destes sentem-se confusos sobre o processo de compilação,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

A COMPILAÇÃO ESTÁTICA E A DINÂMICA

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 da suíte *Mozilla* e do *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 é necessária a 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 uma única mídia de instalação de uma distribuição, já que utilizam este conceito.

AS FERRAMENTAS DO PROJETO GNU

Conforme dissemos anteriormente na *1a. Parte: Os sistemas GNU/Linux - > O Linux*, o *Projeto GNU* consistia de desenvolver um sistema operacional *Unix-like* livre, onde para isto foram construídas inicialmente as principais ferramentas de desenvolvimento. Dentre elas existem desde compiladores, editores, utilitários de automação e uma série de bibliotecas para esta atividade (em especial, o *Emacs*, o *GCC* e *GNU C Library*).

Nesta seção, iremos conhecer todos os componentes indispensáveis para a realização do processo de compilação.

AS BIBLIOTECAS

GNU C LIBRARY

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

A *GNU C Library* - conhecida como *glibc*, ou ainda, *libc6* - é usada por praticamente todos os programas desenvolvidos não só para os sistemas



GNU/Linux, como também para outras plataformas que *Unix*, como o *Hurd*. Ela é essencial para a compilação dos códigos-fonte de seus pacotes.

Esta biblioteca segue as especificações *POSIX*, provendo todos os recursos necessários para o desenvolvimento de aplicações que interajam diretamente com o *kernel* do sistema. Eis porque ela é um componente essencial para a compilação manual de qualquer aplicação.

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 com as versões exigidas da *glibc*, onde raramente tais requisitos são deixados de atender.

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, a manutenção, a portabilidade e o uso de bibliotecas compartilhadas. Ela é indispensável para a compilação das pendências necessárias na instalação de outros programas.

Graças a esta biblioteca, muitas facilidades são também asseguradas para o desenvolvedor, já que este terá certas tarefas simplificadas, como também a eliminação de detalhes mais complexos na resolução de pendências.

AS FERRAMENTAS DE AUTOMAÇÃO

M4

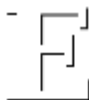
- ✓ <<http://www.gnu.org/software/m4/>>.

O *m4* é a implementação de macro-processadores tradicionais dos ambientes *Unix*. Sua função consiste em auxiliar a geração dos 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*.



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 a 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 especificamente para o sistema em uso.

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*, que por sua vez realiza a compilação dos programas os quais desejamos instalar.

O processo de compilação se dá corretamente graças às instruções de um arquivo especial gerado pelo *configure* - os *Makefiles* -, os quais contém todas as instruções necessárias para a realização do processo.

OS 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 a linguagem *C* e suas variantes (*C++* e *Objective-C*) de acordo com as especificações *ANSI C*, possui excelente performance e portabilidade, além de ter inúmeras outras qualidades, mas que não serão descritas aqui devido ao enfoque da literatura, que é destinada exclusivamente ao usuários *desktops*...



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.

ENTRE OUTRAS...

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.

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 certamente necessitará de uma atualização (senão, o pacote de atualização não existiria...).

Os pacotes de atualização possuem a extensão *.diff*, os quais poderão ser baixados normalmente como um arquivo comum. E antes de utilizá-lo, deveremos copiá-lo para o diretório-raiz onde se encontra o código-fonte do programa original e dentro deste, utilizar as seguintes sintaxes...

Para atualizar um pacote, utilizem:

```
$ 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.

OBTENDO O CÓDIGO-FONTE

Geralmente obtemos o código-fonte dos programas ou *drivers* desejados na página do desenvolvedor. Basta apenas baixá-los e copiá-los para a máquina onde desejamos 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).

DESCOMPACTANDO O 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, abram um terminal e procedam da seguinte forma:

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

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

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

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

Observem que na definição das sintaxes, apenas foram modificadas as opções *z* e *j*, que acionam os compactadores *Bzip2* e *gzip* para realizar a descompressão do pacote agregado pelo *TAR*, respectivamente.

- 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 entrarmos neste diretório...

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

O 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.



A LEITURA DE DOCUMENTAÇÕES

Ao consultarmos a estrutura do diretório criado após a descompactação de um determinado pacote, iremos observar uma estrutura de dados contendo vários arquivos e diretórios, conforme exemplificado a seguir:

```
# 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
# _
```

De acordo com o pacote instalado, veremos inúmeros itens 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, certamente todas as suas particularidades estarão lá descritas, além dos respectivos procedimentos a serem tomados.

A CONFIGURAÇÃO, A COMPILAÇÃO E A INSTALAÇÃO

Para compilar o pacote, existem 3 comandos a serem utilizados:

- *./configure*: script responsável para detecção das pendências e geração de relatórios, chamados *makefile*.



- *make*: ferramenta de automação necessária para a compilação do código-fonte, baseando-se nas instruções do *makefile*.
- *make install*: acionamento da 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 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 ESMTIP 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 ala 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

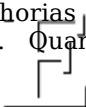
Conforme dito, apesar do corpo da licença não se encontrar, estão descritas as principais cláusulas necessárias para torná-la compatível com a *GPL*.

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...
      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 seqüência 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...
-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.

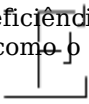
MAKE CLEAN

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

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 na mídia 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.

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.

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



IV. A CONVERSÃO DE PACOTES E PROGRAMAS

INTRODUÇÃO

São diversos os motivos os quais levam os usuários a realizar conversões de pacotes e programas compilados para o formato nativo do *Slackware*, onde o principal deles está na ausência do pacote desejado – lógico!

Neste capítulo, teremos algumas instruções e recomendações necessárias para realizar esta atividade com a indicação e utilização de alguns dos principais utilitários para esta função.

AS FERRAMENTAS...

ALIEN

✓ <http://kitenet.net/programs/alien/>.

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

Para realizar a instalação do *Alien*, no diretório-raiz do programa, digitem...

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

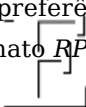
```
# alien [OPÇÕES] [ARQUIVO.FORMATO]
```

Onde:

- *-d (--to-deb)*: converte para o formato nativo do *Debian*;
- *-r (--to-rpm)*: converte para o formato *RPM*;
- *-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
```

Após a conversão, utilizem as ferramentas nativas para realizar a instalação do novo pacote gerado. No caso do *Slackware*...

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

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.

```
# checkinstall
```

```
checkinstall 1.6.0, Copyright 2002 Felipe Eduardo Sanchez Diaz Duran
This software is released under the GNU GPL.
```

A versão mais atual do *CheckInstall* se encontra disponível no FTP da distribuição, mais especificamente na série */extra*. Baixem o pacote e realizem a sua instalação através do comando...

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

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 substituí-lo pela evocação do próprio *CheckInstall*:

```
# checkinstall
```

Assim, automaticamente será criado o pacote desejado, sendo este baseado na respostas de algumas instruções feitas pelo próprio utilitário. Caso queiram criar um pacote sem ter que responder as perguntas feitas pelo *CheckInstall*, 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 a 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
```

Enfatizando novamente, optem sempre por utilizar o *CheckInstall* para empacotar os programas compilados manualmente; assim, poderemos gerenciá-los através do uso das ferramentas nativas do sistema, o que nos possibilitará resolver uma série de problemas.



RPM2TGZ

O utilitário *rpm2tgz* nada mais é do que um conversor de pacotes do *Slackware*, o qual transforma arquivos no formato *RPM* para o seu formato nativo *TGZ*. Sua utilização é bastante simples e prática.

Sintaxe:

```
# rpm2tgz [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 instalar normalmente o pacote no formato *TGZ*.

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

# _
```

Uma situação interessante ocorreu ao removermos este pacote apenas com o uso do comando *removepkg* e o nome do pacote como parâmetro. Embora funcione bem com outros programas, em algumas circunstâncias podem ocorrer alguns problemas. Vejam 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/
```

```
# _
```

Infelizmente o *rpm2tgz* possui alguns inconvenientes de compatibilidade durante a conversão. Por exemplo, não conseguimos rodar a *API Wine* após converter seu pacote pré-compilado no formato *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 então funcionou normalmente.

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 a 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* diferente¹. Se isto já ocorre com os programas compilados no formato nativo de outras versões do *Slackware*, imagine entre distribuições diferentes...

CONCLUSÃO

Por “tradição”, a instalação de programas no *Slackware* é feita diretamente a 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

1 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.



V. OBTENDO OS PACOTES OFICIAIS

INTRODUÇÃO

Diferente das demais distribuições, o *Slackware* mantém uma antiga tradição de que os pacotes externos devem ser instalados a partir da compilação manual. Embora este não seja um método muito prático de se instalar aplicativos, traz algumas vantagens interessantes, como o maior domínio na manutenção da base de pacotes e sua melhor personalização.

Embora existam diversos projetos de repositórios para pacotes interessantes, neste capítulo iremos tratar apenas dos pacotes-base que compõem o repositório oficial da distribuição.

O FTP DO SLACKWARE

✓ [<ftp://ftp.slackware.com/pub/slackware/slackware-current/>](ftp://ftp.slackware.com/pub/slackware/slackware-current/).

É no repositório (*FTP*) do *Slackware* que estão os principais programas utilizados pela distribuição, além de diversos outros pacotes que poderão ser bastante úteis. E os diretórios que compõem a estrutura do *FTP* são:

BOOTDISKS

Em *bootdisks* são armazenadas as imagens de vários *kernels* para a criação de disquetes de inicialização. Cada uma destas imagens possuem uma aplicação específica: por exemplo, normalmente utilizamos a imagem *bare.i* para realizar uma instalação padrão; mas dependendo das circunstâncias, outras imagens poderão ser melhor aplicáveis, como *sata.i* (*HDs SATA*) e *raid.s* (sistemas com arranjos de *HD* em *RAID*).

EXTRA

Apesar dos principais programas estarem inclusos nas mídias de instalação, muitos destes, por questão de espaço e necessidade, não se encontram presentes. Foi criado então o diretório *extra*, que contém os pacotes adicionais e que geralmente são muito úteis para o complemento de funcionalidades ao sistema.

ISOLINUX

Todas as imagens necessárias para a criação de uma mídia para a instalação do sistema. As instruções de como realizar o procedimento ficam armazenadas em um arquivo-texto chamado *README.TXT*.

Ainda neste diretório, temos também o subdiretório *sbootmgr*,



responsável pelo armazenamento da imagem *Smart Boot Manager*. Esta por sua vez é essencial para realizar a instalação do sistema em máquinas que não suporta a inicialização através do *drive* óptico, já que esta imagem deverá ser gravada previamente em um disquete e, com ele, ser feita a inicialização.

KERNELS

Neste diretório estão mantidas uma série de imagens de *kernels* compilados para diversas finalidades. Estas imagens são utilizadas no ato da instalação do sistema, mais especificamente na seção *Configure -> Kernel*.

Seguindo a filosofia das imagens mantidas em *bootdisks*, o mais utilizado é o *bare.i*, que provê suporte aos dispositivos mais comuns. Durante o período em que eram mantidos tanto o *kernel* da série 2.4 (padrão) quanto 2.6, a instalação da versão 2.6 era feita através da imagem *huge26.s*.

PASTURE

Em *pasture* são mantidos pacotes antigos que não fazem mais parte da versão atual do *Slackware*. Para o bom administrador, muitos destes pacotes podem ser úteis, de acordo com as suas necessidades.

PATCHES

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.

ROOTDISK

As imagens do *rootdisk* são necessárias para que seja feita a inicialização do sistema a partir de disquetes. Elas são usadas em conjunto com as imagens disponíveis em *bootdisks*. Atualmente estas imagens são a *install.1* e *install.2*, mas em tempos antigos, eram chamadas por *color.gz*.

SLACKWARE

Os pacotes presentes no diretório principal do *Slackware* - simplesmente "*slackware*" - são os pacotes oficiais da distribuição.

- *a*: os pacotes essenciais para o funcionamento do sistema;
- *ap*: as aplicações em modo texto;
- *d*: interpretadores, compiladores e bibliotecas de desenvolvimento;



- *e*: editor de código *EMACS*;
- *f*: documentações sobre o *Linux* (*Linux-FAQs* e *Linux-HOWTOs*);
- *k*: código-fonte do *kernel*;
- *kde*: ambiente gráfico *KDE* e as aplicações baseadas nele;
- *kdei*: pacotes de internacionalização do *KDE*;
- *l*: bibliotecas extras (necessárias para vários programas);
- *n*: pacotes para trabalhar com redes e seus aplicativos;
- *t*: processador de textos *TEX*;
- *tcl*: pacotes para o desenvolvimento em *TCL/Tk*;
- *x*: o servidor gráfico *X.org*;
- *xap*: aplicativos e interfaces fazem o uso do modo gráfico;
- *y*: jogos *BSD* e derivados.

Eles se encontram disponíveis nas mídias de instalação do sistema, dispensando a sua obtenção via *Internet*. Porém, se necessitarmos de utilizar a versão mais atualizada (*current*), invariavelmente teremos que baixá-los. Para isto, acessem na página oficial do *Slackware*, a seção *mirrors*, para saberem quais são os principais repositórios disponíveis.

SOURCE

Em *source* são mantidos os códigos-fonte das aplicações provenientes desta árvore. Muitas vezes estes pacotes são utilizado por desenvolvedores e usuários mais experientes e que desejam obter ajustes mais finos do sistema com a compilação destes. Para isto, eles editam os *scripts SlackBuilds* presentes junto aos pacotes, já que eles mantêm armazenados os atributos de compilação de cada pacote, além das rotinas necessárias para o seu empacotamento no formato nativo da distribuição (o *TGZ*).

TESTING

Os pacotes que ainda estão em estágio experimental, mas que serão posteriormente integrados ao diretório principal, são mantidos no diretório *testing*, como certos programas não podem ser substituídos pelas versões mais novas “*da noite para o dia*”: o servidor *WEB Apache*, o banco de dados *MySQL*, a linguagem interpretada *PHP* e a versão mais atual do *kernel* são alguns dos mais notáveis exemplos.

As versões antigas - porém estáveis - são mantidas na árvore principal da distribuição, sendo disponibilizadas as versões mais novas neste diretório para aqueles que (por motivos diversos) necessitarem destes pacotes. E claro, ficará por conta e risco do administrador ao adotar tais pacotes.



ZIPSLACK

O *ZipSlack* é uma versão compacta do sistema para que possa ser executado a partir de um *ZIP-drive*, sob um sistema de arquivos no formato *DOS*. É chamado de *ZipSlack* justamente por causa do espaço ocupado, já que são consumidos mais que *100 MB*. E claro, todos os arquivos, pacotes e imagens de inicialização são mantidos neste diretório.

Embora todo o conteúdo caiba em apenas *100 MB*, são necessários pelo menos *300 MB* livres e disponíveis para que possamos utilizar o sistema, segundo as suas instruções *README.1st*.

SOBRE A ÁRVORE /CURRENT

✓ [<ftp://ftp.slackware.com/pub/slackware/slackware-current/>](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 a versão atualizada dos programas que compõem a distribuição, este é um local ideal para dar o pontapé inicial.

CHECANDO A INTEGRIDADE DOS PACOTES

Dependendo da conexão utilizada, muitas vezes o conteúdo dos arquivos e pacotes baixados da *Internet* não se encontram íntegros, face às possibilidades de interferências na linha telefônica durante a realização do processo, entre outros possíveis problemas. Para checar a integridade destes arquivos, utilizarem 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, existem 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 [PACOTE]
```

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 igual a fornecida, isto significa que seu pacote se encontra íntegro; caso contrário, ocorreu algum erro do qual resultou no corrompimento do pacote baixado. Neste último caso será necessário baixá-lo novamente.

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

