Crie pacotes oficiais de software para o Slackware

Embrulha pra presente!

No último artigo, aprendemos a gerenciar pacotes no Slackware e até nos aventuramos a criar alguns com o auxílio do utilitário checkinstall. Neste artigo, vamos estudar como os pacotes são montados na distribuição principal, desbravando os SlackBuilds. POR SULAMITA GARCIA

SlackBuilds são scripts que automatizam a criação de pacotes no formato oficial do Slackware. São *shell scripts* (para saber mais sobre shell scripts, veja a série "Papo de Botequim" , na página 87 desta edição) que, apesar de padronizados, variam ligeiramente de acordo com as necessidades do software a ser instalado. Criar um SlackBuild facilita muito a tarefa de manutenção de futuras versões do pacote.

Os SlackBuilds estão disponíveis em qualquer servidor ftp que contenha a árvore de pacotes do Slackware. Nessa árvore há o diretório *source*, que contém o código-fonte de tudo que compõe o Slackware. Dentro do diretório de cada pacote há seu código-fonte, um arquivo com a descrição e um SlackBuild, no formato *nome-do-software.slackbuild*.

Para demonstração, vamos utilizar um pacote simples, como o *irssi*. Ele esta disponível no servidor oficial (*ftp://ftp.slackware.com/slackware-10.0/source/n/irssi/irssi.SlackBuild*). Além de simples, ele é um pacote que foi incluído há pouco tempo na distribuição, com definições mais recentes.

Como é um shell script, o SlackBuild do irssi começa com a definição do shell e de algumas variáveis:

```
#!/bin/sh
CWD=`pwd`
TMP=${TMP:-/tmp}
PKG=$TMP/package-irssi
```

Estas linhas indicam o *sh* como shell, definem a variável *CWD* (*Current Work Directory* - Diretório Atual de Trabalho) como o diretório atual, a variável *TMP*, se já não estiver definida, aponta para */tmp*, e *PKG* recebe o valor *TMP/package-nome-do-software*, em nosso caso, *irssi*. Logo a seguir vêm as definições das variáveis de compilação. Veja a Listagem 1.

Desde o Slackware 9, se a variável *ARCH* (arquitetura) não estiver definida, os pacotes são compilados para a arquitetura 486. Se estiver, o script vai definir algumas flags de compilação e guardá-las dentro da variável *SLKCFLAGS*. Posso otimizar o código para minha máquina, que tem um processador Athlon XP, trocando *mcpu = i386* por *mcpu = athlon-xp*:

```
if [ ! -d $TMP ]; then
 mkdir -p $TMP
fi
rm -rf $PKG
mkdir -p $PKG
cd $TMP
rm -rf irssi-$VERSION
tar xjvf $CWD/irssi-$VERSION.tar.bz2
```

Aqui o script verifica se o diretório temporário definido na variável *TMP* existe e remove eventuais resíduos de uma compilação anterior. Em seguida, o código-fonte é descompactado neste diretório.

```
cd irssi-$VERSION
chown -R root.root .
find . -perm 777 -exec chmod 755 {} \;
find . -perm 664 -exec chmod 644 {} \;
```

Após entrar no diretório do código-fonte, o script realiza uma verificação de permissões, retirando as permissões de

Listagem 1

```
VERSION=0.8.9
ARCH=${ARCH:-i486}
BUILD=3

if [ "$ARCH" = "i386" ]; then
       SLKCFLAGS="-O2 -march=i386 -mcpu=i686"
elif [ "$ARCH" = "i486" ]; then
       SLKCFLAGS="-O2 -march=i486 -mcpu=i686"
elif [ "$ARCH" = "s390" ]; then
       SLKCFLAGS="-O2"
elif [ "$ARCH" = "x86_64" ]; then
       SLKCFLAGS="-O2"
fi
```

acesso dos usuários ao código-fonte e dando acesso apenas para o *root*. A seguir vem a configuração do pacote propriamente dita:

```
CFLAGS="$SLKCFLAGS" \
./configure \
--prefix=/usr \
--sysconfdir=/etc \
--enable-ipv6 \
--with-textui \
--with-bot \
--with-proxy
make
make install DESTDIR=$PKG
```

As opções de compilação são passadas ao gcc através da variável *CFLAGS*. Aqui você pode adicionar ou remover opções: se você não quer suporte a *ipv6*, remova a linha correspondente. Se quiser testar a *garbage collection*, deve adicionar uma linha especificando isso (*--with-gc* \). Para uma lista com todas as opções do pacote, digite *./configure -help*.

Depois de tudo isso, o pacote é compilado e os binários, páginas de manual, documentação e demais arquivos serão instalados no diretório especificado na variável *DESTDIR*, como se estivessem na raiz do sistema. Se você observar o que foi instalado, notará que os diretórios */usr*, */etc* e outros foram criados em DESTDIR.

Na Listagem 2, o dono e o grupo dos binários são modificados para usuário *root* e grupo *bin*. Em seguida o script procura os binários no formato ELF e realiza um *strip*. Isto vai "limpar" o executável, removendo símbolos de depuração, o que diminui o tamanho dos arquivos. Logo após, o script muda a extensão do arquivo de configuração para *.new*, para não sobrescrever algum arquivo de configuração já existente durante a instalação. Continuando nossa análise, veja o código na Listagem 3.

Lá o script faz alguns ajustes na documentação, movendo o que realmente interessa para um diretório com um nome significativo, composto pelo nome e versão do pacote.

```
gzip -9 $PKG/usr/man/man?/*.?
```

A linha acima comprime as páginas de manual (padrão no Slackware, lembra?).

```
mkdir -p $PKG/install
cat $CWD/slack-desc > $PKG/install➥
/slack-desc
zcat $CWD/doinst.sh.gz > $PKG➥
/install/doinst.sh
```

Aqui são criados os arquivos *slack-desc* e *doinst.sh*, discutidos no primeiro artigo. Recapitulando: o *slack-desc* é um arquivo de descrição do pacote e *doinst.sh* é um shell script que irá realizar tarefas pós-instalação, como a criação de links, novos usuários e ajustes necessários ao bom funcionamento do pacote.

A localização desses arquivos pode ser modificada e o *zcat* pode ser substituído pelo *cat*, caso o arquivo *doinst.sh* seja um shell script sem compactação.

```
cd $PKG
makepkg -l y -c n $TMP/irssi-➥
$VERSION-$ARCH-$BUILD.tgz
```

Finalmente podemos criar o pacote com o *makepkg*. As opções significam:

- *-l y*: se um link simbólico é encontrado, uma instrução para criá-lo é adicionada ao script *doinst.sh*. Em seguida, o link é removido.
- *-c n*: se não for *n* (no), o makepkg irá configurar todas as permissões de acesso ao diretório para *755* e o dono para *root.root*. Não é o que queremos.

Agora você já sabe como funciona um script que cria um pacote para o Slackware. Que tal utilizar esse conhecimento para "empacotar" um software que ainda não faz parte da distribuição? ■

Listagem 2

```
chown -R root.bin $PKG/usr/bin
( cd $PKG
  find . | xargs file | grep "executable" | grep ELF | cut -f1 -d : | ➥
xargs strip --strip-unneeded 2> /dev/null
  find . | xargs file | grep "shared object" | grep ELF | cut -f 1 -d : | ➥
xargs strip --strip-unneeded ➥ 2> /dev/null
)
mv $PKG/etc/irssi.conf.$PKG /etc/irssi.conf.new
```

Listagem 3

```
rm -rf $PKG/usr/share/doc
( cd $PKG
  find . -name perllocal.pod | xargs rm -f
)
 mkdir -p $PKG/usr/doc/irssi-$VERSION
 cp -a AUTHORS COPYING INSTALL NEWS README TODO docs $PKG/usr/doc/irssi-➥
$VERSION
 rm -f $PKG/usr/doc/irssi-$VERSION/docs/Make* ➥ $PKG/usr/doc/irssi-➥
$VERSION/docs/*.1
```

Erramos!

No artigo "Embrulha pra Viagem!", na página 55 de nosso primeiro número, um erro de edição prejudicou a compreensão de um trecho do texto. Após correção, o primeiro parágrafo da página 55, fica:

O arquivo do-install.sh também não é específico ao Slackware. Ele é genérico, então para criar um pacote no formato oficial devemos substituí-lo pelo doinst.sh. Não realizar estas alterações não implica no mau funcionamento do pacote, só significa que ele não está de acordo com o padrão. É neste momento que você pode incluir tarefas no shell script doinst.sh. Nele você pode executar comandos no sistema, adicionar usuários, criar links, etc.

SOBRE A AUTORA

Sulamita Garcia é formada em Ciências da Computação pela UFSC, onde conheceu o Linux. Participa do projeto LinuxChix e mantém o site de Alta Disponibilidade na UnderLinux, possuindo também certificação LPIC II. Trabalha atualmente na Cyclades como Software Designer.