# Anatomia de um Slackbuild

#### Preâmbulo

Acho que todos os usuários do Slackware já usaram um script do SlackBuild para criar um pacote de software que pode ser instalado de forma fácil e limpa e removido posteriormente, se necessário. Minha experiência foi que eu executei o script SlackBuild para criar um pacote e nunca examinei o código que ele continha até que um dia a versão de origem não correspondesse ao indicado no script.

Por que você se incomodaria em olhar o código em um script SlackBuild? Bem, primeiro você pode usá-lo se estiver aprendendo bash. Em segundo lugar, talvez você não queira enviar um script do SlackBuild, mas talvez você queira criar um pacote que não existe para o software desejado.

Então, aqui eu gostaria de passar por um script do SlackBuild da melhor maneira possível (verbalmente). Congratulo-me abertamente com edições de todos os outros em todos os níveis. Na verdade, acho importante que pessoas de todos os níveis contribuam. O que é óbvio para uma pessoa pode precisar de explicação para outra pessoa.

Existem diferentes abordagens para um script do SlackBuild e, por exemplo, Alien Bob tem seu criador muito útil do SlackBuild em : Aliens SlackBuild Toolkit

Atualmente, no SlackBuilds.org, eles preferem envios usando um modelo do SlackBuild. Eu acho que seria útil passar por um script do SlackBuild que está atualmente no SlackBuilds.org. Para isso, usarei o script SlackBuild que enviei para latex2html latex2html para o SlackBuilds.org e atualmente disponível para o Slackware 14.2.

Dessa forma, estamos falando de um script SlackBuild de trabalho contemporâneo que você pode realmente baixar, usar e, com sorte, se relacionar depois de ler isso.

A outra coisa a mencionar é que o **context** desta página, quando você o lê, sobre um script latex2html é que todo um “slackbuild” do slackbuilds.org foi baixado, é um local conveniente (digamos na minha área de trabalho) que ele está descompactado e no diretório descompactado chamado “latex2html”, a fonte do software chamada latex2html-2019.2.tar.gz foi colocada.

Além disso, você iniciou o script:

```
bash-5.0# chmod a+x latex2html.SlackBuild
bash-5.0# ./latex2html.SlackBuild
```

#### Bash

Nos dias anteriores ao Windows no Unix, um shell ou Bourne Shell (Stephen Bourne, Bell Labs) era uma maneira de se comunicar com o sistema. Em um aceno para Stephen Bourne, Brian Fox surgiu com uma nova versão em 1989 e a chamou de **B**ourne **a**gain **Sh**ell (bash).

Se você abrir um terminal no Slackware - diga `Menu → Sistema → Console` e digite no prompt `$`:

```
bash --version
```

, você deveria ter algo como:

```
GNU bash, version 5.0.11(1)-release (x86_64-slackware-linux-gnu)
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
```

Em termos simples, a entrada foi recebida e adotada. A entrada é um pouco curta, pois quando você fecha a janela, a entrada desapareceu.

Um script bash em termos simples é o código bash que pode ser salvo como um arquivo no disco rígido.

A primeira linha de um script bash possui esta linha:

```
#!/bin/sh
```

Isso geralmente é chamado de *shebang*. Isso define que o arquivo é um script de shell e também o caminho para o interpretador de shell.

### SlackBuild Script

Até agora, descrevemos que um script do SlackBuild é, em essência, o código de apresentação apresentado em uma apresentação mais permanente de um arquivo.

No topo está o *shebang*.

Depois disso, de acordo com os requisitos do slackbuilds.org, as próximas linhas são o nome do pacote, quem escreveu o script e, opcionalmente, um aviso de direitos autorais.

Tentarei percorrer a linha 14.2 latex2html slackbuild por linha. Se houver espaços em branco, é um aviso para que outros participem. Provavelmente também significa que eu não entendo completamente essa parte.

```
PRGNAM=latex2html
VERSION=${VERSION:-2019.2}
BUILD=${BUILD:-1}
TAG=${TAG:-_SBo}
```

Se eu olhar o pacote que instalei no meu Slackware de 64 bits atual, é: `latex2html-2019.2-x86_64-1_SBo`. Agora, se você olhar para a primeira linha acima, verá PRGNAM (nome do programa). Essa é uma variável e seu valor é definido como a string `latex2html`.

A versão está relacionada à versão da fonte do software; se você for ao fonte do lates você verá que houve um lançamento de fonte em 5 de junho de 2019. Assim, simplesmente nomeei a versão após esse lançamento. Para quem está aprendendo bash (ainda estou): na segunda linha VERSION= à esquerda atribui um valor à variável chamada VERSION.

Ao atribuir um valor a VERSION, por que usar `VERSION=${VERSION:-2019.2}` e não simplesmente`VERSION=2019.2`?

Pegue a expressão `${VERSION}`. Isso significa “o valor contido na variável VERSION”. A notação mais complexa `${VERSION:-2019.2}` significa “o valor contido na variável VERSION, mas se essa variável ainda não tiver um valor, use o valor padrão de '2019.2'”.
Portanto, a segunda linha simplesmente se resume a `VERSION=2019.2`.
Um valor para VERSION pode ser definido fora do script: se você especificou um valor para `VERSION` na linha de comando do SlackBuild ou se foi definido no seu ambiente Bash.

Da mesma forma, a variável TAG é definida como SBo e quando o pacote é criado, isso mostra que é um pacote “slackbuild”. Se você olhar para pacotes em outros repositórios, verá no nome, por exemplo `chromium-77.0.3865.75-x86_64-1alien` que o pacote é do Alien Bob.

O próximo bloco de código é o seguinte:

```
if [ -z "$ARCH" ]; then
  case "$( uname -m )" in
    i?86) ARCH=i586 ;;
    arm*) ARCH=arm ;;
    *) ARCH=$( uname -m ) ;;
  esac
fi
```

Nas linguagens de programação de computadores, como php, python e outras, incluindo o bash, existem práticas e lógica comuns. Um princípio comum é o procedimento de realizar um teste no código. Um teste é aplicado e, é claro, haverá um resultado dependendo do resultado. Um exemplo de A declaração do bloco bash “if” é:

```
if [ test condition ]
then
    código a ser executado dependendo do resultado
fi
```

No bloco acima, simplesmente o “se” é o início de uma condição de teste If e “fi” indica o final de uma condição de teste If.

Voltando ao bash por um minuto, você pode definir uma variável do bash assim

```
ARCH=alguma_coisa
```

e você pode obter o valor de uma variável e ver qual o valor que a variável “ARCH” contém, colocando um cifrão na frente assim:

```
echo $ARCH
```

Um sinalizador -z pode ser usado em uma instrução “if” para ver se uma sequência está vazia. Então, vamos dar uma olhada na primeira linha novamente e descobrir o que isso significa.

```
if [ -z "$ARCH" ]; then
```

Acima, não precisamos ver o valor de ARCH; contanto que possamos acessar seu valor e poder usá-lo no script. Portanto, a primeira parte antes do “então” simplesmente equivale a, se o valor da variável

ARCH for expresso como uma string e seu valor estiver vazio, faça alguma coisa. ARCH pode representar um valor de seqüência da arquitetura do PC no qual o script está sendo executado.

Outro algoritmo comum é chamado, por exemplo, php é a “instrução switch”. Isso posto em seus termos mais simples é uma lista e o teste é para ver se um determinado valor que você possui pode ser encontrado nessa lista. Se não for encontrado, você pode colocar na última linha da lista que gostaria de fazer, se nada corresponder.

No bash, em princípio, é a mesma idéia, mas é chamada de “case statement” .Em princípio, você pode colocar o que quiser no case statement, mas se pensar no fato de que queremos descobrir a “arquitetura” de um computador e já sabemos que existem apenas algumas possibilidades, faz sentido experimentar primeiro algumas das opções conhecidas na lista e, se não houver correspondência, faça algo para obter uma resposta. arm e i586 são obviamente dois tipos de arquitetura de computadores.

Agora, se você tentar esse código em uma janela de terminal:

```
uname -m
```

ele deve exibir a arquitetura do seu PC, no meu caso, o x86_64.

Então, para resumir sobre o bloco de código. Primeiro, uma instrução “if” é executada para verificar se a variável “ARCH” está vazia. Se houver um valor para a variável ARCH, nada no bloco “if” e “case interno” será executado; mas se uma string vazia for encontrada (ARCH não tem valor), uma “instrução case” será executada (dentro do if bloco de código) para encontrar uma correspondência. Se uma correspondência fosse encontrada, a variável ARCH seria configurada com o valor da correspondência encontrada e a execução do caso seria interrompida. Se a variável ARCH estava vazia e nenhuma correspondência foi encontrada na lista de casos, a linha inferior vem int play, que é:

```
uname -m
```

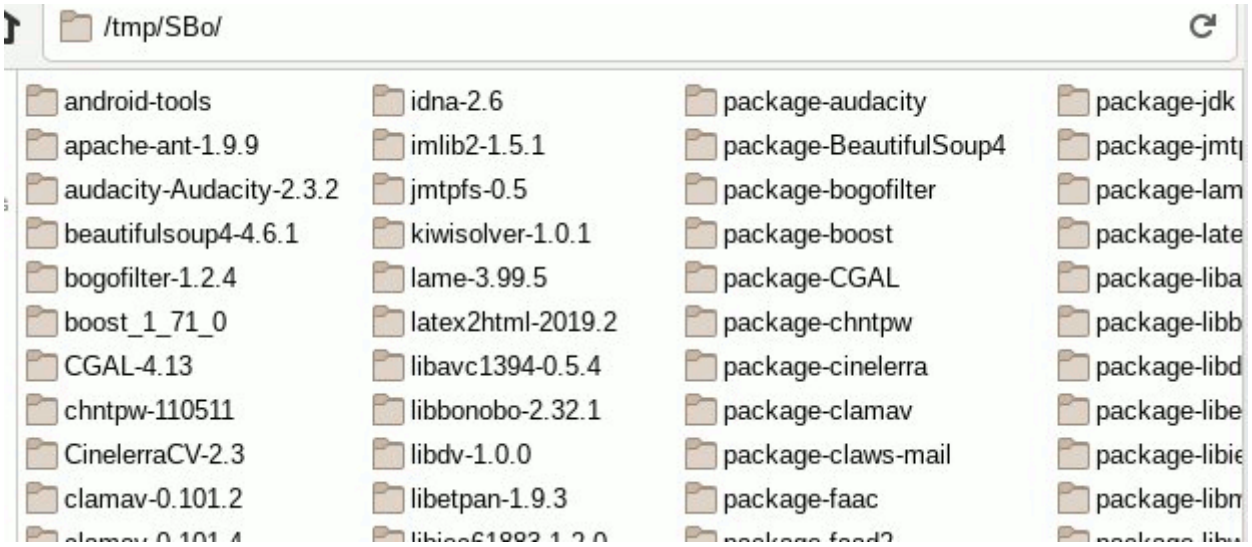
é usado para obter um resultado e ARCH definido para o resultado.

Não se assuste com o ponto de interrogação no i?86, o ponto de interrogação é um espaço reservado que permite possibilidades via regex. pode ser “3” (i386), “6” (i686) etc.

Próximo bloco de código:

```
CWD=$(pwd)
TMP=${TMP:-/tmp/SBo}
PKG=$TMP/package-$PRGNAM
OUTPUT=${OUTPUT:-/tmp}
```

Antes de entrarmos nisso, vamos dar uma olhada no meu sistema de arquivos do slackware e ver o que há em /tmp/SBo. Fazendo uma rápida olhada na imagem, você terá uma pista de que o slackbuild funciona, usando o /tmp/SBo/ diretório e cria um diretório com a sintaxe package-packagename. Se agora dermos uma olhada no código acima. CWD (diretório de trabalho atual) é uma variável e é definido como o valor de pwd. Se você executar isso em uma janela de terminal, ele informará onde você está no contexto do bash de onde está trabalhando.




O TMP será definido como /tmp/SBo. Agora vamos dar uma olhada

```
PKG=$TMP/package-$PRGNAM
```

Você pode imaginar que o PKG será definido para latex2html como:

```
/tmp/SBo/package-latex2html
```

Se você olhar atentamente para a imagem (levando em consideração /tmp/SBo/ na parte superior da imagem), verá exatamente isso na imagem.OUTPUT está definido como /tmp

Próximo bloco de código:

```
if [ "$ARCH" = "i586" ]; then
  SLKCFLAGS="-O2 -march=i586 -mtune=i686"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O2 -march=i686 -mtune=i686"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2 -fPIC"
  LIBDIRSUFFIX="64"
else
  SLKCFLAGS="-O2"
  LIBDIRSUFFIX=""
fi
```

Provavelmente precisamos, antes de examinarmos o restante do código, o latex2html slackbuild, para introduzir alguns conceitos básicos.

Historicamente, o software de computador é instalado em um processo de três etapas chamado configure, make, make install. O Configure se prepara para criar o software, verificar se tudo está disponível e criar um novo arquivo make. Um arquivo make é um arquivo que contém instruções para criar um programa.

Na linha de comando, você pode instalar o software apenas usando configure, make e make install.

Um slackbuild faz o trabalho de criar um pacote para que o processo de instalação seja mais gerenciável e confiável. Quando você faz o download de um slackbuild em seu sistema, deve haver uma nova versão do código-fonte; basta colocar essa fonte no slackbuild descompactado e editar rapidamente o script do slackbuild.

A fonte do programa de computador é escrita em linguagens de “alto nível”, mas é convertida em um formato que o computador possa entender prontamente. O código escrito na linguagem C envolve um compilador C que o converteu em binário.

O objetivo de qualquer sistema que esteja instalando um programa é que ele envolva o conceito de torná-lo “feito sob medida” para o computador que está sendo instalado. Obviamente, isso envolverá a arquitetura do computador. Durante o processo de compilação, o sistema pode ser ajustado passando opções de variáveis.

Vamos agora dar uma olhada no bloco de código acima. O bloco de código é simplesmente um bloco “if, else”, onde o código é executado de cima para baixo e equivale a -se a arquitetura é i586, defina SLKFLAGS como .. se não vá para a próxima linha.

CFLAGS e CXXFLAGS são variáveis contendo valores que podem ser passados no tempo de compilação. Veremos mais adiante que a variável SLKFLAGS será usada para defini-las.

Próximo bloco de código:

```
set -e
rm -rf $PKG
mkdir -p $TMP $PKG $OUTPUT
cd $TMP
rm -rf $PRGNAM-$VERSION
tar xvf $CWD/$PRGNAM-$VERSION.tar.gz
cd $PRGNAM-$VERSION
chown -R root:root .
find -L . \
 \( -perm 777 -o -perm 775 -o -perm 750 -o -perm 711 -o -perm 555 \
  -o -perm 511 \) -exec chmod 755 {} \; -o \
 \( -perm 666 -o -perm 664 -o -perm 640 -o -perm 600 -o -perm 444 \
  -o -perm 440 -o -perm 400 \) -exec chmod 644 {} \;
```

set -e: isso interrompe a execução do script se houver um erro ao executar este código após este comando
rm -rf $PKG: isso exclui qualquer diretório (e conteúdo) anterior do package-latex2html em/tmp/SBo/package-latex2html
mkdir -p $TMP $PKG $OUTPUT : O mkdir com o sinalizador “-p” cria diretórios, mas somente se eles já não existirem.

isso seria /tmp/SBo , /tmp/Sbo/package-latex2html, /tmp

É improvável que o diretório SBo não exista, a menos que nenhum outro slackbuilds tenha sido executado no passado. /tmp deve estar lá como padrão na instalação do slackware.
cd $tmp : move o local onde o bash está trabalhando para /tmp/SBo

rm -rf $PRGNAM-$VERSION irá se livrar de quaisquer entradas anteriores do diretório (talvez falhe) para latex2html-2019.2

tar xvf $CWD/$PRGNAM-$VERSION.tar.gz : Isso equivale a descompactar latex2html-2019.2.tar.gz, que estaria dentro do slackbuild descompactado do slackbuilds.org “latex2html”.

cd $PRGNAM-$VERSION : Este é um comando “alterar diretório”. O shell bash agora estará funcionando a partir do contexto localizado dentro da fonte descompactada, localizada em $CWD. ou seja, estamos de volta ao slackbuild descompactado, mas dentro da fonte descompactada.

chown -R root:root . : Observe o ponto, com um espaço no final da linha; isso significa tudo no diretório atual. -R é permissão recursiva. Então, aqui estamos dando propriedade ao root e grupo root.

```
 find -L . \
 \( -perm 777 -o -perm 775 -o -perm 750 -o -perm 711 -o -perm 555 \
  -o -perm 511 \) -exec chmod 755 {} \; -o \
 \( -perm 666 -o -perm 664 -o -perm 640 -o -perm 600 -o -perm 444 \
  -o -perm 440 -o -perm 400 \) -exec chmod 644 {} \;
```

Se eu entendi o bloco de código acima corretamente, use o “find” com base nas permissões e siga os links simbólicos usando o “-L flag”. Se eu entendi isso corretamente, basicamente defini os diretórios para 755, a fim de ativar um “cd” neles e nos arquivos para que o root possa ler a gravação.

```
-type d -exec chmod 775 {}
```

Para diretório

```
-type f -exec chmod 644 {}
```

Para arquivos.

Próximo bloco de código:

```
CFLAGS="$SLKCFLAGS" \
CXXFLAGS="$SLKCFLAGS" \
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --sysconfdir=/etc \
  --localstatedir=/var \
   --with-perl=/usr/bin/perl \
  --enable-eps \
  --enable-gif \
  --enable-png \
  --build=$ARCH-slackware-linux \
  --host=$ARCH-slackware-linux

make
make install DESTDIR=$PKG
```

Algumas coisas a dizer aqui, o uso de “\” é uma maneira de usar um comando longo em várias linhas. Já mencionamos CFLAGS e CXXFLAGS. Também mencionamos anteriormente a variável SLKFLAGS e aqui a usamos para definir o valor de CFLAGS e CXXFLAGS.

Nesse script latex2html slackbuild, também utilizamos um processo de três etapas para o configure, make, make install. Mas e quanto a –enable-eps, de onde isso vem?

Bem, se você pegar o código fonte latex2html source descompacte-o (clique rápido, abra com o Ark) no CD e execute:

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

Então você vai ter alguma informação útil dos desenvolvedores. Ele mostra a opção e como você pode habilitar alguns deles.

```
make
make install DESTDIR=$PKG
```

Aqui o, make install é realizado. Nota DESTDIR é uma flag para dizer para onde o pacote vai ser instalado

$PKG equivale a /tmp/SBo/package-latex2html

Próximo bloco de código:

```
find $PKG -print0 | xargs -0 file | grep -e "executable" -e "shared object"
| grep ELF \
  | cut -f 1 -d : | xargs strip --strip-unneeded 2> /dev/null || true

mkdir -p $PKG/usr/doc/$PRGNAM-$VERSION
cp -a \
 FAQ INSTALL LICENSE MANIFEST README.md TODO  \
  $PKG/usr/doc/$PRGNAM-$VERSION
cat $CWD/$PRGNAM.SlackBuild > $PKG/usr/doc/$PRGNAM-
$VERSION/$PRGNAM.SlackBuild
cp $CWD/manual.pdf  $PKG/usr/doc/$PRGNAM-$VERSION

mkdir -p $PKG/install
cat $CWD/slack-desc > $PKG/install/slack-desc

cd $PKG
/sbin/makepkg -l y -c n $OUTPUT/$PRGNAM-$VERSION-$ARCH-
$BUILD$TAG.${PKGTYPE:-tgz}
```

As duas primeiras linhas desse bloco são um pouco complicadas.

```
find $PKG -print0 | xargs -0 file | grep -e "executable" -e "shared object"
| grep ELF \
  | cut -f 1 -d : | xargs strip --strip-unneeded 2> /dev/null || true
```

No entanto, podemos escolher as palavras-chave que são comandos e que podem ajudar a entender isso. ind “localizado em /usr/bin/find é um utilitário poderoso que possui cerca de 50 opções. Ele basicamente faz o que diz na lata. Com a opção -print0, ele separa o que encontra com “\000”- em uma palavra NULL .

O “|” ou pipe é usado para passar os resultados de um comando para outro; nesse caso, xargs que possui um sinalizador -0. Essa opção é para o xargs aceitar a entrada que possui /000 entre eles. ELF é é executável e formato vinculável.

Para dar uma resposta sucinta, as duas linhas estão removendo símbolos de depuração e outras coisas desnecessárias para tornar os binários menores, mais rápidos e ocupar menos memória.

```
mkdir -p $PKG/usr/doc/$PRGNAM-$VERSION
```

Aqui estamos preparando um diretório que será chamado “latex2html-2019.2” localizado em /usr/doc. Assim, colocamos a documentação relevante em um diretório chamado de forma relevante, para que um usuário possa acessar a documentação no pacote. As próximas linhas colocam arquivos como README.md dentro do diretório /usr/doc/latex2html-2019.2.

```
cat $CWD/$PRGNAM.SlackBuild > $PKG/usr/doc/$PRGNAM-
$VERSION/$PRGNAM.SlackBuild
```

Essa linha volta ao diretório original do qual o Latex2html.SlackBuild foi executado (citei anteriormente o Desktop) abre o SlackBuild com o comando “cat” e o copia para o diretório de documentação. Antes de enviar o latex2html para o slackbuilds, obviamente fiz alguns testes e descobri que quando o pacote foi instalado, havia uma saída bastante abrangente do que poderia ser feito usando:

```
$ latex2html --help
```

Também tive acesso a um manual abrangente em formato pdf; portanto, no meu caso, não escrevi código para páginas de manual. Em vez disso, basta colocar uma cópia do “manual.pdf” no diretório /usr/doc/latex2html-2019.2.

*— andy brookes 2020/01/05 16:29 (UTC)*

*If you teach maths it doesn't stop you embedding a little English.*

Modelos de slackbuild em branco podem ser obtidos em : https://slackbuilds.org/templates/

# Fontes

Estou usando um script SlackBuild que enviei para slackbuilds.org:latex2htmlslackbuild

- Originalmente escrito por andy brookes
- *— slackjeff 2020/01/14 18:31 (BRT)*

howtos