Empacotamento Debian usando schroot e cowbuilder

David da Silva Polverari

Introdução

schroot é um utilitário que:

- facilita o gerenciamento de vários ambientes chroot
- possibilita o uso desses ambientes por usuários comuns, diferente do chroot normal
- pode trabalhar com diretórios como chroot comum ou de outras maneiras (arquivos compactados, arquivos loopback, dispositivos de bloco, ou snapshots LVM e BRTFS.
- possibilita montar automaticamente sistemas de arquivos importantes dentro deles (através de scripts de configuração, alguns já providos)
- permite alterar os ambientes chroot durante o uso e, ao terminar, deixá-lo como estava originalmente.

Estas características favorecem a sua utilização no trabalho de empacotamento Debian, pelos seguintes motivos:

- simplifica o trabalho de criação de um ambiente chroot para empacotamento, através da automatização provida pelos scripts de configuração do schroot.
- facilita a manutenção e atualização dos ambientes chroot originais em sua forma inalterada.
- diminui a necessidade de utilização do usuário root.
- evita a necessidade de sincronização de arquivos de configuração e variáveis de ambiente entre o ambiente nativo do empacotador e os chroots, através da montagem automática do diretório home do usuário e da preservação de seu ambiente.
- facilita o uso de múltiplos ambientes chroot (sid, stable, etc)

Premissas

Esse documento não é destinado a iniciantes na atividade de empacotamento. Se considera que o leitor está familiarizado com o uso de jaulas chroot para empacotamento, conforme descrito nos documentos e vídeos criados e disponibilizados por Eriberto Mota em http://www.debianet.com.br.

Além disso, este documento considera e demonstra o seguinte cenário:

- O usuário possui a configuração dos programas relacionados com o empacotamento em seu diretório home (dotfiles).
- O usuário configurou `sudo` de acordo com o descrito na página respectiva do Debian Wiki.

Formas de acesso e namespaces

schroot permite a utilização de um ambiente chroot de três formas distintas:

- acesso ao ambiente original (namespace *source:*), no qual todas as modificações realizadas são mantidas.

- acesso a um ambiente temporário, utilizando *copy-on-write* ou união de sistemas de arquivos tendo como base o ambiente source, cujas modificações são descartadas ao deixar o ambiente (namespace *chroot:*).
- criação de uma sessão com a utilização dos mesmos mecanismos utilizados pelo namespace *chroot:*, entranto no qual as modificações serão mantidas até a finalização explícita da sessão, o que permite continuar o trabalho mesmo após sair do ambiente.

Instalação e configuração

Nesta seção, vamos descrever a configuração e a utilização do schroot para a criação de ambientes chroot temporários destinados ao empacotamento Debian, e do cowbuilder e pbuilder para construção final do pacote em um ambiente totalmente limpo, para confirmar o trabalho realizado.

Exemplos de arquivos de configuração

Nesta seção temos exemplos de arquivos de configuração que serão compartilhados tanto pelas ferramentas que executarão no ambiente nativo do empacotador (cowbuilder e pbuilder, por exemplo), quanto por aquelas que estarão executando dentro dos ambientes chroot. Configure de acordo com suas necessidades.

As configurações apresentadas aqui foram adaptadas a partir das recomendações apresentadas no Guide for Debian Maintainers (<https://www.debian.org/doc/manuals/debmake-doc/ch03.en.html>) e no Guia Rápido de Empacotamento de Software no Debian (pocket), versão 3.3 (<http://eriberto.pro.br/debian/guia_empacotamento_debian_3.3.pdf>).

Neste exemplo foram colocados os arquivos que estão mais diretamente relacionados com o empacotamento, porém todos os arquivos de configuração no diretório home (dotfiles) serão compartilhados com os ambientes chroot.

Primeiro adicione essas linhas a seu .bashrc:

~/.bashrc

```
# Debian env vars for devscripts
DEBEMAIL="david.polverari@gmail.com"
DEBFULLNAME="David da Silva Polverari"
export DEBEMAIL DEBFULLNAME

# Alias for Debian packaging-specific quilt config
_completion_loader quilt
alias uscan-check="uscan --verbose --report"
alias dquilt="quilt --quiltrc=${HOME}/.quiltrc-dpkg"
complete -F _quilt_completion $_quilt_complet_opt dquilt
```

~/.config/lintian/lintianrc

```
display-info = yes
pedantic = yes
display-experimental = yes
color = auto
```

~/.devscripts

```
DEBUILD_DPKG_BUILDPACKAGE_OPTS="-i -I -us -uc"
DEBUILD_LINTIAN_OPTS="-i -I -E --pedantic --show-overrides --color auto"
DEBSIGN_KEYID="1D01384528E2C751597ADE13024F09655CEB233F"
```

~/.dput.cf

```
[mentors]
fqdn = mentors.debian.net
incoming = /upload
method = https
allow_unsigned_uploads = 0
progress_indicator = 2
# Allow uploads for UNRELEASED packages
allowed_distributions = .*
```

~/.gbp.conf

```
[DEFAULT]
# Uncomment below to use cowbuilder/pbuilder
# builder = git-pbuilder -i -I -us -uc
debian-branch = debian/master
pristine-tar = True
color = auto
```

~/.gitconfig

```
[user]
    name = David Polverari
    email = david.polverari@gmail.com
[core]
    editor = vim
```

~/.pbuilderrc

```
# options from https://www.debian.org/doc/manuals/debmake-doc/ch03.en.html
AUTO_DEBSIGN="${AUTO_DEBSIGN:-no}"
PDEBUILD_PBUILDER=cowbuilder
HOOKDIR="/var/cache/pbuilder/hooks"
MIRRORSITE="http://deb.debian.org/debian/"
APTCACHE=/var/cache/apt/archives
BUILDRESULT=../
EXTRAPACKAGES="ccache lintian libeatmydata1"

# enable to use libeatmydata1 for pbuilder
#export LD_PRELOAD=${LD_PRELOAD+$LD_PRELOAD:}libeatmydata.so

# enable ccache for pbuilder
#export PATH="/usr/lib/ccache${PATH+:$PATH}"
#export CCACHE_DIR="/var/cache/pbuilder/ccache"
#BINDMOUNTS="${CCACHE_DIR}"

# parallel make
#DEBBUILDOPTS=-j8
```

~/.quiltrc-dpkg

```
d=.
while [ ! -d $d/debian -a `readlink -e $d` != / ];
    do d=$d/..; done
if [ -d $d/debian ] && [ -z $QUILT_PATCHES ]; then
    # if in Debian packaging tree with unset $QUILT_PATCHES
    QUILT_PATCHES="debian/patches"
    QUILT_PATCH_OPTS="--reject-format=unified"
    QUILT_DIFF_ARGS="-p ab --no-timestamps --no-index --color=auto"
    QUILT_REFRESH_ARGS="-p ab --no-timestamps --no-index"
    QUILT_COLORS
="diff_hdr=1;32:diff_add=1;34:diff_rem=1;31:diff_hunk=1;33:diff_ctx=35:diff_cctx=33"
    if ! [ -d $d/debian/patches ]; then mkdir $d/debian/patches; fi
fi
```

/var/cache/pbuilder/hooks/B90lintian

```
#!/bin/sh
set -e
apt-get -y --allow-downgrades install lintian
echo "+++ lintian output +++"
su -c "lintian -i -I -E --pedantic --show-overrides --color auto
/tmp/buildd/*.changes; :" -l pbuilder
echo "+++ end of lintian output +++"
```

NOTE	O arquivo `/var/cache/pbuilder/hooks/B90lintian` deve receber permissão de execução para outros: `sudo chmod o+x /var/cache/pbuilder/hooks/B90lintian`

Instalando os pacotes necessários no host

Instale os pacotes debootstrap e schroot:

```
sudo apt install cowbuilder debootstrap pbuilder schroot
```

Inicializando o ambiente chroot

Primeiro, crie o diretório que servirá como base para o chroot utilizando debootstrap:

```
sudo debootstrap sid /srv/chroot/sid http://deb.debian.org/debian
```

Configuração do schroot para utilizar o ambiente criado

A configuração do schroot pode ser realizado através do arquivo de configuração `/etc/schroot/schroot.conf`, ou de maneira modular, através de arquivos `*.conf` no diretório `/etc/schroot/chroot.d`.

Crie o arquivo de configuração `sid.conf` dentro do diretório `/etc/schroot/chroot.d` para configurar a utilização do ambiente chroot criado anteriormente:

/etc/schroot/chroot.d/sid.conf

```
[sid]
description=Debian sid (unstable)
type=directory ①
directory=/srv/chroot/sid
users=polverari ②
source-users=polverari ③
preserve-environment=true ④
union-type=overlay ⑤
```

Essa configuração irá configurar um ambiente denominado `sid` do tipo diretório, localizado em `/srv/chroot/sid`, cujos ambientes `chroot:` e `source:` poderão ser acessados pelo usuário comum `polverari`, e suas variáveis de ambiente serão preservadas dentro do chroot, o que permitirá, entre outras coisas, a execução de programas X11 sem a necessidade de configurações adicionais.

Os ambientes chroot dentro dos namespaces `chroot:` e `session:` serão fornecidos baseados na união de sistemas de arquivos provida pelos kernels Linux 4.0+.

Realizando a configuração base do ambiente chroot para construção de pacotes

Primeiramente, entre no ambiente chroot source para que as modificações sejam permanentes:

```
schroot -c source:sid -u root
```

Dentro do ambiente source, instale os pacotes necessários:

```
apt install autopkgtest bash-completion blhc devscripts dh-make dput-ng git \
git-buildpackage lintian locales quilt spell splitpatch sudo tree vim wget
```

NOTE	A instalação de pacotes no ambiente chroot criado pelo debootstrap poderia ser realizada através da opção `--include=` desse comando. Essa abordagem foi evitada, entretanto, uma vez que o debootstrap tem problemas com resolução de dependências de pacotes virtuais, o que provoca o término prematuro da criação do ambiente, deixando-o em um estado inconsistente (#878961).

Adicione uma linha `deb-src` ao arquivo `/etc/apt/sources.list`:

/etc/apt/sources.list

```
deb http://deb.debian.org/debian sid main
deb-src http://deb.debian.org/debian sid main
```

Depois atualize o cache do APT:

```
apt update
```

Configure os locales desejados, descomentando as linhas respectivas no arquivo `/etc/locale.gen` e execute `locale-gen` para gerar os locales:

```
locale-gen
```

Por último, remova um dpkg-statoverride causado por um bug do schroot (#565613), e que pode impedir a instalação de novos pacotes:

```
dpkg-statoverride --remove /etc/exim4/passwd.client
```

Saia do ambiente source:

```
exit
```

Usando schroot para empacotamento

Nesta seção mostramos um exemplo do uso de schroot para o trabalho de empacotamento.

Listando os ambientes chroot:

Para listar os ambientes chroot disponíveis:

```
schroot -l
```

O comando seguinte mostra inclusive as sessões ativas:

```
schroot -l -a
```

Configurando ou atualizando um ambiente source

Para configurar, adicionar pacotes ou atualizar um ambiente chroot original (source), podemos acessá-lo através do namespace `source:`:

```
schroot -c source:sid
sudo apt update && sudo apt upgrade -y
```

Alternativamente, é possível entrar como usuário root no ambiente:

```
schroot -c source:sid -u root
apt update && apt upgrade -y
```

Acessando o ambiente chroot regular

Para acessar um ambiente chroot que terá suas modificações descartadas ao sair, podemos acessá-lo da seguinte maneira:

```
schroot -c sid
```

Ao entrar no ambiente, se realiza normalmente os trabalhos de empacotamento, saindo dele com o comando `exit`.

Todas as modificações (instalação de pacotes, alterações de arquivos, etc) realizadas no chroot

serão descartadas, à exceção dos arquivos em /home. Desse modo, é possível manter os pacotes em um subdiretório de home (ex: `~/pkg`) e realizar os trabalhos de empacotamento dentro do ambiente chroot.

Utilizando sessões

Para trabalhos de empacotamento demorados, é possível usar sessões, que funcionam de forma semelhante aos ambientes chroot regulares, porém suas alterações somente são descartadas após o encerramento explícito da sessão, permitindo retomar os trabalhos em um momento posterior.

Iniciando uma sessão

Para iniciar uma sessão, se utiliza a opção `-b` (*begin session*). Para facilitar o gerenciamento, aconselha-se nomear a sessão com a opção `-n`:

```
schroot -b -c sid -n sessao_pacote
```

A partir desse momemnto, o comando `schroot -l -a` irá listar a sessão como ativa.

Entrando ou retornando a uma sessão

A entrada ou retorno à uma sessão é realizada com a opção `-r` (*run session*):

```
schroot -r -c sessao_pacote
```

A partir desse momento, o empacotador pode realizar seu trabalho e interrompê-lo temporariamente a qualquer hora, saindo do ambiente com o comando `exit`. As modificações não serão perdidas.

Terminando a sessão

Quando a sessão não for mais necessária, ela pode ser terminada com a opção `-e` (*end session*):

```
schroot -e -c sessao_pacote
```

Utilizando cowbuilder para construir o pacote final

Esses passos devem ser executados no ambiente nativo do empacotador.

Criando um ambiente para o cowbuilder

Caso não possua ainda um ambiente para a construção do pacote no cowbuilder, esse deve ser

criado através do seguinte comando:

```
sudo -E cowbuilder create
```

Como o cowbuilder necessita privilégios de root para executar, utilizamos comando `sudo -E` para manter o ambiente do usuário comum e usar o arquivo de configuração em seu diretório home, evitando assim a cópia ou criação de link simbólico do arquivo de configuração no diretório `/root`.

Atualizando o ambiente do cowbuilder

Para atualizar o ambiente do cowbuilder, basta executar o comando:

```
sudo -E cowbuilder update
```

Construção do pacote

No diretório do upstream, execute o comando:

```
pdebuild
```