

[Creare pacchetti per SLACKWARE – Tecniche e Approfondimenti](#)

By nEcROSoFt

[Introduzione e scopo del documento](#)

“Creare pacchetti è un'arte” e come tale ognuno la interpreta come preferisce e come crede. Non esiste, infatti, un unico modo di creare pacchetti per Slackware ma si possono adottare varie metodologie più o meno automatizzate a seconda dei gusti e delle necessità. Ovviamente ogni strada è buona sotto il comune denominatore della corretta riuscita del pacchetto, quindi “arte sì” ma fatta bene. In questo articolo esporrò il mio modo di crearli che potrà piacere come no, in ogni caso per chi si avvicina alla pacchettizzazione può essere un buon punto di riferimento per iniziare. Il metodo che verrà presentato è “ibrido”, cioè una metodologia semi-automatizzata dettata dall'esperienza che prevede l'utilizzo di uno *SlackBuild* (vedremo di cosa si tratta...) e di configurazioni manuali. *Ovviamente, per una più rapida e migliore comprensione dell'articolo è richiesto un minimo di conoscenza di bash scripting.*

[I Pacchetti](#)

Con il termine “pacchetto” si identifica quell'insieme di file che costituiscono un Software che è stato preventivamente compilato in modo tale da poter posizionare questi file in modo corretto all'interno del sistema. *Non ci sarà necessità di compilare nulla (intendo in fase di installazione del pacchetto...) proprio perchè il pacchetto contiene già software precompilato.* Il nostro scopo sarà, quindi, quello di compilare sorgenti facendo attenzione ai permessi, ai parametri di compilazione e ad una serie di aspetti che concorreranno alla realizzazione di un buon “subgenius package”.

[Introduzione alla struttura di Slackware](#)

Prima di iniziare a vedere come si possa realizzare effettivamente un pacchetto è bene fornire una visione di insieme della struttura di Slackware facendo attenzione agli aspetti stilistici che la contraddistinguono da sempre.

Struttura delle directory:

```
drwxr-xr-x root root ./
drwxr-xr-x root root ../
drwxr-xr-x root bin bin/
drwxr-xr-x root root boot/
drwxr-xr-x root root dev/
drwxr-xr-x root root etc/
drwxr-xr-x root root home/
drwxr-xr-x root root lib/
drwx----- root root lost+found/
drwxr-xr-x root root mnt/
drwxr-xr-x root root opt/
dr-xr-xr-x root root proc/
drwx--x--- root root root/
```

```
drwxr-xr-x root bin sbin/  
drwxr-xr-x root root sys/  
drwxr-xr-x root root temp/  
drwxrwxrwt root root tmp/  
drwxr-xr-x root root usr/  
drwxr-xr-x root root var/
```

La cosa fondamentale da notare sono i gruppi e i proprietari delle varie directory. La gran parte è *root:root* ma *sbin* e *bin* risultano avere come gruppo *bin* (*root:bin*).

Si dovrà fare molta attenzione in seguito ai permessi e alle proprietà delle directory dei nostri pacchetti.

Si noti, infine, che praticamente tutte le directory *bin* e *sbin* del sistema (es. */usr/bin* o */usr/sbin*) hanno come gruppo *bin* (e ovviamente come proprietario *root*).

Il “Subgenius Style”

La “tradizione” di Slackware prevede dei canoni standard per andare a posizionare nel sistema i file di un pacchetto necessari al funzionamento di un programma.

Directory per la documentazione: */usr/doc/*

Directory che tengono traccia di pacchetti e scripts: */var/log/*

Directory per i binari */usr/bin* - */usr/sbin*

Directory per le librerie: */usr/lib*

Directory per i files di configurazione: */etc* (in “alternativa” anche */usr/etc*)

E' bene ricordare anche le seguenti directories per il mantenimento delle informazioni sui pacchetti e script aggiunti/rimossi.

/var/log/removed_scripts

/var/log/packages

/var/log/removed_packages

/var/log/scripts

N.B. Si noti che la gran parte delle distribuzioni (Redhat, Fedora, etc...) prevede, invece, il posizionamento dei binari in */usr/local/bin*, delle librerie in */usr/local/lib* etc...

Strumenti per la gestione dei pacchetti

Per gestire i pacchetti si useranno i seguenti strumenti:

<i>pkgtool</i>	<i>(Completa gestione dei pacchetti con interfaccia testuale.)</i>
<i>makepkg</i>	<i>(Creazione automatizzata di un pacchetto partendo da un precompilato.)</i>
<i>explodepkg</i>	<i>(Estrae il pacchetto nella directory corrente.)</i>
<i>installpkg</i>	<i>(Installa il pacchetto.)</i>
<i>removepkg</i>	<i>(Rimuove il pacchetto.)</i>
<i>upgradepkg</i>	<i>(Disinstalla il vecchio pacchetto e ne installa uno nuovo.)</i>

La compilazione e il processo di installazione

Vediamo adesso come avviene il processo di compilazione e di installazione di un programma a partire dai suoi sorgenti per poter realizzare un pacchetto.

Il nostro scopo è quello di installare un programma all'interno di una directory "fittizia" che "emuli" '/' in modo da poter avere sotto controllo tutte le directorys e i files che un programma utilizza. A tal scopo è necessario modificare, in fase di configurazione/compilazione, le directorys in cui il programma andrà ad installarsi.

Possiamo usare il flag `--prefix`. Posizionandoci all'interno della directory contenente i files sorgenti possiamo lanciare `configure` in questo modo:

```
$ ./configure --prefix=/usr
```

In questo modo possiamo cambiare il path di installazione. Ovviamente il sorgente deve essere dotato di "configure" (man `autoconf` vi può aiutare...). Se sono necessari files in `/etc/` utilizzeremo:

```
$ ./configure --sysconfdir=/etc
```

Esistono altri flags per altre directory, potete farvi un'idea digitando: `./configure --help`.

E' necessario considerare anche un aspetto di ottimizzazione durante la configurazione. I pacchetti standard per Slackware (si fa riferimento allo standard per i pacchetti dalla Slackware9.0 in poi) sono compilati per architetture i486. Dobbiamo quindi utilizzare flags necessari all'ottimizzazione dei sorgenti per questa architettura. Un comando tipico che possiamo impartire è il seguente:

```
$ CFLAGS="-O2 -march=i486 -mcpu=i686" ./configure --prefix=/usr
```

Successivamente possiamo compilare normalmente digitando:

```
$ make
```

E dopo installare il tutto con:

```
$ make prefix='percorso'/usr install
```

o anche:

```
$ make DESTDIR='percorso'/usr install
```

Dove 'percorso' sarà il nostro path personalizzato per installare il software in una specifica directory temporanea (cioè la dir che emula '/' come dicevo in precedenza). Spostandovi in questa directory, infatti, vi sembrerà di essere sotto '/'. Troverete directorys come `/usr` e `/etc` che conterranno altre directorys come `/usr/bin` o `/usr/lib`. Da qui potremmo creare il nostro package.

Ok, fino a qui dovremmo avere un'idea in generale abbastanza chiara di quali strumenti sono necessari per realizzare un pacchetto. Prima di vedere l'effettiva procedura **COMPLETA** (fino ad ora si è cercato di fissare i caratteri generali...) di creazione di un pacchetto, dobbiamo esplicitare altri standard necessari.

Files & Standard per i Packages

Il nome dei pacchetti

Un pacchetto dovrà avere un nome di questo tipo:

nomeprg-versioneprg-architettura-build.tgz

es. **ddclient-0.6.3-i486-1.tgz**

In alternativa si possono aggiungere tre lettere dopo 'build' che rappresentano l'identificativo del creatore del pacchetto. Per esempio un mio package avrà un nome del tipo: **ddclient-0.6.3-i486-1nec.tgz** dove 'nec' è il mio identificativo (da nEcrOSoFt).

Lo slack-desc

Installando pacchetti avrete sicuramente notato una descrizione del programma dopo aver lanciato 'installpkg nomefile.tgz'. Tale descrizione va inserita proprio nel file *slack-desc*. Esiste uno standard per questo file, potete semplicemente copiare ed incollare lo *slack-desc* generico che riporto successivamente personalizzandolo con le informazioni del programma che state pacchettizzando. **slack-desc:**

```
<-----CUT HERE----->
# HOW TO EDIT THIS FILE:
# The "handy ruler" below makes it easier to edit a package description. Line
# up the first '|' above the ':' following the base package name, and the '|'
# on the right side marks the last column you can put a character in. You must
# make exactly 11 lines for the formatting to be correct. It's also
# customary to leave one space after the ':'.

|----handy-ruler-----|
bind: bind version (DNS server and utilities)
bind:
bind: Description...
bind: Description...
bind: Description...
bind: Description...
bind:
bind:
bind: Package created by YOUR NAME HERE
bind:
bind:
<-----CUT HERE----->
```

ATTENZIONE: notate che il primo carattere '|' (sopra bind) deve risultare *esattamente* sotto i due punti. Inoltre le righe (quelle che portano il tag 'bind') devono essere esattamente 11 per una corretta formattazione. La voce 'bind' dovrà essere ovviamente sostituita con il 'nomeprg' nella sezione “*il nome dei pacchetti*”.

[Il doinst.sh](#)

In questo file si può inserire qualsiasi comando eseguibile in shell. Viene utilizzato qualora si abbia necessità di eseguire comandi all'installazione del pacchetto. Ad esempio potreste voler dare permessi particolari a qualche file (sconsigliato, ma a volte può essere utile) o voler appendere alla fine di uno script delle linee di scripting, etc...

ATTENZIONE: E' bene notare che questo file risulta molto importante all'esecuzione del comando 'makepkg' (che vedremo fra poco...) visto che talvolta (in presenza di librerie in particolar modo o anche documentazione...) vengono inserite in questo file automaticamente delle direttive (*symlinks*).

[La cartella 'install'](#)

Lo *slack-desc* e il *doinst.sh* dovranno essere posizionati in questa cartella che poi sarà “letta” in fase di installazione del package per poter eseguire i comandi presenti in *doinst.sh* e per poter visualizzare la descrizione del programma contenuta nello *slack-desc*. Si tratta di una cartella speciale posizionata all'interno della directory temporanea di pacchettizzazione che conterrà il precompilato. *La sua creazione può essere automatizzata come vedremo dallo Slackbuild.*

[Lo Slackbuild](#)

Ora possiamo finalmente vedere la procedura per la creazione effettiva di un pacchetto. Come detto in “*Introduzione e scopo del documento*” verrà utilizzato un metodo ibrido: in parte automatizzato da uno SlackBuild e in parte manuale. **Con Slackbuild si intende semplicemente uno script che automatizzi la creazione di un pacchetto.** Nello Slackbuild riportato successivamente sono presenti anche dei commenti per esemplificare alcune scelte.

Slackbuild:

```
<-----CUT HERE----->
#!/bin/sh
# SlackBuild by nEcROSoFt.
# Contrariamente a quanto fanno molti io uso cartelle all'interno della home per creare
# pacchetti.
CURRENT=/home/user/~dir~ #Posizionare qui slack-desc, doinst.sh e SlackBuild.
TMP=/home/user/~dir~/PackageBuild #Campo variabile. Creare la cartella.
NAME=nomeprg #Campo variabile.
VERSION=version #Campo variabile.
ARCH=i486 #Campo variabile.
BUILD=1 #Campo variabile.
PKG=$TMP/package-$NAME #Campo variabile. La cartella e' creata dallo script.
if [ ! -d $PKG ]; then
    mkdir -p $PKG
```

```

fi
echo "*-----*"
echo " ##  ## ##### ##### ## ## ## "
echo " ### ## ## ## ## ## ## "
echo " ## # ## ##### ## ##### ## ## "
echo " ## ### ## ## ## ## ## "
echo " ## ### ##### ##### ## ## ## "
echo "*-----SlackBuild---*"
echo "< $NAME-$VERSION >"
echo "#-----#"
#Scompattare in 'PackageBuild' il tarball.
cd $TMP/$NAME-$VERSION
# Io lancio 'configure' prima a mano con: 'CFLAGS="-O2 -march=i486 -mcpu=i686"
#!/configure --prefix=/usr' per verificare i makefile. Potrebbero aver bisogno di modifiche
#manuali.
make
make prefix=$PKG/usr install
mkdir -p $PKG/usr/doc/$NAME-$VERSION
cp -a \
AUTHORS BUGS COPYING ChangeLog INSTALL NEWS README \
$PKG/usr/doc/$NAME-$VERSION
strip $PKG/usr/bin/*
#Le pagine man preferisco gzipparle a mano successivamente.
chown -R root:bin $PKG/usr/bin
chmod 644 $PKG/usr/doc/$NAME-$VERSION/*
chown -R root:root $PKG/usr/doc/$NAME-$VERSION
chown -R root:root $PKG/usr/share
mkdir -p $PKG/install
cat $CURRENT/slack-desc > $PKG/install/slack-desc
cat $CURRENT/doinst.sh > $PKG/install/doinst.sh
#cd $PKG
#makepkg -ly -c n $TMP/$NAME-$VERSION-$ARCH-$BUILD.tgz
#Decommentando le due righe precedenti create immediatamente il pacchetto, ma in
#generale io preferisco farlo a mano dopo averne controllato l'effettiva correttezza.

<-----CUT HERE----->

```

Una volta creato uno Slackbuild di questo tipo, per poterlo lanciare dobbiamo dare allo script i permessi di esecuzione. Possiamo farlo digitando:

```
$ chmod a+x Slackbuild
```

La procedura

Prima di lanciare Slackbuild dobbiamo preparare l'ambiente per la creazione del pacchetto:

1. Posizionare *slack-desc* e *doinst.sh* (se non fosse necessario inserire comandi all'interno

del *doinst.sh* potete non crearlo, lo farà in ogni caso lo script per voi) nella cartella che preferite. C'è chi preferisce creare i pacchetti in */tmp*, chi preferisce farlo all'interno di una cartella nella propria home. **Il path è indicato dalla variabile CURRENT.**

2. Creare la directory che conterrà sia la cartella dei sorgenti, sia la cartella del software precompilato. **Il path è indicato dalla variabile TMP.**
3. Scompattare in **TMP** il tarball contenente i sorgenti.
4. Accertarsi della presenza di “*configure*” e lanciarlo come abbiamo visto in precedenza:

```
$ CFLAGS="-O2 -march=i486 -mcpu=i686" ./configure --prefix=/usr
```

5. Verificare i/il Makefile generati per accertarsi preventivamente del supporto di “*prefix*”. Troverete, infatti, nei Makefile una riga: “*prefix*”. Se alla fine del vostro procedimento di creazione il pacchetto non risultasse corretto (ad esempio cartelle */usr/bin* o */usr/lib* mancanti), significa che nel Makefile non si verifica “propagazione del path di installazione”. Questo vuol dire che alcune path (per le librerie, o magari per i binari...) sono state inserite staticamente dall'autore del tarball e il parametro “*prefix*” non comporta nessun cambiamento dei path di installazione. Con molta pazienza, quindi, dovrete aprire uno ad uno i vari Makefile (se ce n'è uno solo tanto meglio...) e inserire A MANO la path di installazione desiderata (cioè la path indicata nello script da: *\$PKG/usr*).

6. Lanciare lo Slackbuild (da root ovviamente):

```
$ ./Slackbuild
```

7. Al termine del lavoro dello Slackbuild (più o meno lungo a seconda delle dimensioni del software) spostarsi in *PKG* per verificare la corretta presenza dei precompilati e per gzippare le pagine man. Se il software prevede la loro installazione troverete nella vostra dir di pacchettizzazione temporanea una cartella man (*\$PKG/usr/man/man(Z)/* dove *Z* indica il livello delle pagine man: 1,2,3..etc.). Per gzippare queste pagine eseguite all'interno della cartella man(*Z*) il comando:

```
$ gzip *
```

8. Possiamo finalmente creare il package spostandoci in *PKG* e digitando:

```
$ makepkg -l y -c n nomeprg-versioneprg-architettura-build.tgz
```

Le due opzioni hanno il seguente significato:

```
-l, --linkadd y|n (inserisce i symlinks nel file doinst.sh: raccomandato)  
-c, --chown y|n (setta tutti i permessi a root:root / 755: NON raccomandato)
```

Ovviamente se siete sicuri che il pacchetto non avrà errori potete decommentare le ultime righe dello Slackbuild e crearlo automaticamente. **ATTENZIONE:** Ricordatevi che così facendo, senza aggiungere ulteriori comandi allo Slackbuild, le pagine man non saranno

grippate!

Ulteriori precisazioni & Permessi

Copia della documentazione

```
cp -a \  
AUTHORS BUGS COPYING ChangeLog INSTALL NEWS README \  
$PKG/usr/doc/$NAME-$VERSION
```

Con questo comando vengono copiati i file riguardanti le informazioni del software nella cartella indicata: `$PKG/usr/doc/$NAME-$VERSION`. Prima di lanciare lo Slackbuild assicuratevi di averlo modificato aggiungendo (o levando) i vari file di documentazione presenti nel sorgente (ad esempio potreste trovare altri file come: README.pings, ABOUT-NLS, etc...).

Strip

```
strip $PKG/usr/bin/*
```

E' un comando di ottimizzazione. Si esegue sui binari e ne riduce le dimensioni su disco.

I permessi

```
chown -R root:bin $PKG/usr/bin  
chmod 644 $PKG/usr/doc/$NAME-$VERSION/*  
chown -R root:root $PKG/usr/doc/$NAME-$VERSION  
chown -R root:root $PKG/usr/share
```

Con questi comandi viene settata la totalità dei permessi sui file del precompilato. Notiamo nuovamente (per l'ultima volta...) il gruppo *bin* per i binari. E' buona prassi ricontrollare tutti i permessi del precompilato prima di creare effettivamente il pacchetto. Posizionandovi nella vostra directory dei precompilati (*PKG*) potete digitare:

```
$ ls -alR
```

Conclusioni

Trattare la creazione dei pacchetti in modo “esauriente” è impossibile in un unico articolo e ritengo sia impossibile trattarla anche in 2,3...n articoli. Questo perchè ogni pacchetto necessita, molto spesso, di attenzioni particolari. Qualche pacchetto dovrà essere realizzato completamente a mano, altri necessiteranno della directory */sbin*, altri ancora vi faranno impazzire per cercare tutte le “path statiche” all'interno dei/del Makefile.

ATTENZIONE:

E' bene ricordare, infine, che è molto importante la versione del compilatore con cui si compila il sorgente, nonchè la versione delle glibc che può causare il malfunzionamento di qualche pacchetto.

[Risorse](#)

Potete trovare pacchetti per slackware su:

<http://www.linuxpackages.net/>

<http://www.slacky.it/>

<http://necrosoft.altervista.org>

Articolo realizzato da Dario Maggiari aka *nEcROSoFt*

Home Page: <http://necrosoft.altervista.org>

Mail to: necrosoft@slacky.it