RPM-HOWTO

Donnie Barnes, <djb@redhat.com> prevedel Andrej Grauf <andrej.grauf@club.win-ini.si> v2.0, 8. april 1997, prevod 17. december 1998

Kazalo

1 Uvod

RPM je kratica za paketni upravljalnik podjetja Red Hat (ang. **R**ed Hat **P**ackage **M**anager) . Čeprav v imenu vsebuje Red Hat, je popolnoma odprt sistem za pakiranje, ki ga lahko vsakdo uporablja. Uporabnikom omogoča zajetje izvorne kode nove programske opreme, ki jo lahko zapakirajo v izvorno in binarno obliko, tako da lahko binarno kodo enostavno namestijo in ji sledijo, izvorno kodo pa lahko enostavno preoblikujejo. Prav tako vzdržuje bazo podatkov za vse pakete in njihove datoteke, ki se uporabljajo za preverjanje paketov in iskanje informacij o datotekah in/ali paketih.

Podjetje Red Hat Software vzpodbuja ostale prodajalce distribucij, naj si ogledajo RPM in ga uporabijo za svoje distribucije. RPM je zelo prilagodljiv in enostaven za uporabo, čeprav nudi osnovo za zelo obsežne sisteme. Prav tako je popolnoma odprt in vsem na voljo. Zelo veseli bomo vaših obvestil o napakah in njihovih popravkih. RPM je dovoljeno uporabljati in razširjati brez dobička pod pogoji GPL.

Podrobnejša dokumentacija o RPM je na voljo v knjigi avtorja Eda Baileya, Maximum RPM. Knjigo lahko zajamete ali naročite na `<http://www.redhat.com/>`.

2 Pregled

Najprej mi dovolite, da navedem nekaj ozadja o RPM. Eden izmed namenov razvoja je bil, omogočiti uporabo „prvotne“ izvorne kode. Pri RPP (naš prejšnji sistem za pakiranje, s katerim RPM nima nič skupnega), je bila izvorna koda iz katere smo gradili, "razbita" izvorna koda. Teoretično je možno namestiti izvorni RPP in ga potem brez težav zgraditi. Ampak izvorna koda ni bila organizirana in nikjer ni bilo navedeno, kakšne spremembe je potrebno narediti, da se bo paket zgradil. Najprej je potrebno ločeno zajeti prvotno izvorno kodo. Z RPM imate prvotno izvorno kodo skupaj s popravki, s pomočjo katerih smo prej prevajali. Nam se zdi to velika prednost. Zakaj? Iz praktičnih razlogov. Eden je v tem, da vam ob izidu nove verzije programa ni potrebno vedno začeti od začetka, da bi ga prevedli pod RHL. Ogledate si lahko popravek in tako ugotovite, kaj boste verjetno morali postoriti. Na ta način lahko privzete nastavitve za prevajanje enostavno pregledate.

RPM je oblikovan tako, da ima močne informacijske izbire. Pakete lahko iščete v celotni bazo podatkov ali pa samo v določenih datotekah. Prav tako lahko enostavno ugotovite, h kateremu paketu datoteka spada in od kod prihaja. RPM datoteke so stisnjeni arhivi, vendar lahko posamezne pakete hitro in enostavno preiščete, ker je paketu običajno dodana binarna glava, z vsemi možnimi informacijami, ki bi jim morali vedeti. To omogoča hitro iskanje.

Prav tako močna lastnost je sposobnost preverjanja paketov. Če ste v skrbeh, da ste izbrisali za nek paket pomembno datoteko, lahko to preverite. O morebitnih nepravilnostih boste obveščeni. Če je potrebno, lahko sedaj paket ponovno namestite. Ohranijo se tudi vse vaše nastavitvene datoteke.

Za mnogo drugih idej in konceptov, ki so vključeni v RPM, bi se radi zahvalili ljudem pri distribuciji BOGUS. Čeprav je bil RPM v celoti napisan v Red Hat Software, njegovo delovanje temelji na kodi, ki so jo napisali pri BOGUS-u

(PM in PMS).

3 Splošne informacije

3.1 Pridobitev RPM

Najboljši način, da pridete do RPM je, da namestite Red Hat Linux. Če tega ne želite, lahko RPM vseeno dobite in uporabljate. Najdete ga lahko na `<ftp://ftp.redhat.com/pub/redhat/code/rpm>`.

3.2 Zahteve za RPM

Minimalna zahteva za uporabo RPM je `cpio` 2.4.2 ali novejša verzija. Ta sistem je namenjen za uporabo z Linux-om, vendar bo z veliko verjetnostjo deloval tudi na drugih Unix sistemih. Dejansko je bil preveden na SunOs, Solaris, AIX, Irix, AmigaOS in drugih. Ampak pazite, binarni paketi, tvorjeni na drugih tipih Unixa ne bodo združljivi.

To so minimalne zahteve za namestitev RPM-jev. Za izgradnjo RPM-jev iz vira potrebujete tudi vse programe, ki so potrebni za izgradnjo paketov, kot npr. `gcc`, `make`, itd.

4 Uporaba RPM

V najenostavnejši obliki, se za namestitev paketov lahko RPM uporabi v obliki:

```
rpm -i foobar-1.0-1.i386.rpm
```

Naslednji enostaven ukaz je za odstranitev paketa:

```
rpm -e foobar
```

Eden izmed celovitejših, ampak zelo uporabnih ukazov vam omogoča namestitev paketov preko FTP-ja. Če ste povezani z omrežjem in bi želeli namestiti nov paket, morate podati samo veljavni URL naslov, nekako takole:

```
rpm -i ftp://ftp.pht.com/pub/linux/redhat/rh-2.0-beta/RPMS/foobar-1.0-1.i386.rpm
```

Ne pozabite, da se bo ta RPM sedaj preverjal in/ali nameščal preko FTP-ja.

Ker so to preprosti ukazi, se lahko rpm uporablja, kot je razvidno iz sporočila o uporabnosti (Usage massage), na različne načine:

```
RPM version 2.3.9
Copyright (C) 1997 - Red Hat Software
This may be freely redistributed under the terms of the GNU Public License

usage: rpm {--help}
       rpm {--version}
       rpm {--initdb}   [--dbpath <dir>]
       rpm {--install -i} [-v] [--hash -h] [--percent] [--force] [--test]
                        [--replacepkgs] [--replacefiles] [--root <dir>]
                        [--excludedocs] [--includedocs] [--noscripts]
                        [--rcfile <file>] [--ignorearch] [--dbpath <dir>]
                        [--prefix <dir>] [--ignoreos] [--nodeps]
                        [--ftpproxy <host>] [--ftpport <port>]
```

```
                 file1.rpm ... fileN.rpm
rpm {--upgrade -U} [-v] [--hash -h] [--percent] [--force] [--test]
                 [--oldpackage] [--root <dir>] [--noscripts]
                 [--excludedocs] [--includedocs] [--rcfile <file>]
                 [--ignorearch]  [--dbpath <dir>] [--prefix <dir>]
                 [--ftpproxy <host>] [--ftpport <port>]
                 [--ignoreos] [--nodeps] file1.rpm ... fileN.rpm
rpm {--query -q} [-afpg] [-i] [-l] [-s] [-d] [-c] [-v] [-R]
                 [--scripts] [--root <dir>] [--rcfile <file>]
                 [--whatprovides] [--whatrequires] [--requires]
                 [--ftpuseport] [--ftpproxy <host>] [--ftpport <port>]
                 [--provides] [--dump] [--dbpath <dir>] [targets]
rpm {--verify -V -y} [-afpg] [--root <dir>] [--rcfile <file>]
                 [--dbpath <dir>] [--nodeps] [--nofiles] [--noscripts]
                 [--nomd5] [targets]
rpm {--setperms} [-afpg] [target]
rpm {--setugids} [-afpg] [target]
rpm {--erase -e} [--root <dir>] [--noscripts] [--rcfile <file>]
                 [--dbpath <dir>] [--nodeps] [--allmatches]
                 package1 ... packageN
rpm {-b|t}[plciba] [-v] [--short-circuit] [--clean] [--rcfile  <file>]
                 [--sign] [--test] [--timecheck <s>] specfile
rpm {--rebuild} [--rcfile <file>] [-v] source1.rpm ... sourceN.rpm
rpm {--recompile} [--rcfile <file>] [-v] source1.rpm ... sourceN.rpm
rpm {--resign} [--rcfile <file>] package1 package2 ... packageN
rpm {--addsign} [--rcfile <file>] package1 package2 ... packageN
rpm {--checksig -K} [--nopgp] [--nomd5] [--rcfile <file>]
                    package1 ... packageN
rpm {--rebuilddb} [--rcfile <file>] [--dbpath <dir>]
rpm {--querytags}
```

Podrobnosti o uporabi teh stikal lahko najdete na strani o RPM za `man`.

5 No, kaj pa lahko *dejansko* počnem z RPM?

RPM je zelo uporabno orodje in, kot lahko vidite, ima kar nekaj stikal. Njihova uporabnost je najlažje razvidna iz primerov. Kako enostavno namestiti in odstraniti pakete, sem navedel že zgoraj, sedaj pa si oglejmo še nekaj drugih primerov:

- Predpostavimo, da ste po nesreči zbrisali nekaj datotek, niste pa prepričani, kaj ste zbrisali. Če želite preveriti vaš celoten sistem in ugotoviti, kaj bi lahko manjkalo, uporabite

  ```
  rpm -Va
  ```

- Predpostavimo, da naletite na nepoznano datoteko. Da bi ugotovili, kateremu paketu pripada, uporabite:

  ```
  rpm -qf /usr/X11R6/bin/xjewel
  ```

  Rezultat bi bil:

  ```
  xjewel-1.6-1
  ```

- Našli ste koules RPM, pa ne veste kaj to je. Da bi izvedeli kaj o njem, uporabite:

  ```
  rpm -qpi koules-1.2-2.i386.rpm
  ```

Rezultat, ki bi ga dobili:

```
Name        : koules                    Distribution: Red Hat Linux Colgate
Version     : 1.2                       Vendor: Red Hat Software
Release     : 2                         Build Date: Mon Sep 02 11:59:12 1996
Install date: (none)                    Build Host: porky.redhat.com
Group       : Games                     Source RPM: koules-1.2-2.src.rpm
Size        : 614939
Summary     : SVGAlib action game with multiplayer, network, and sound support
Description : This arcade-style game is novel in conception and excellent in
execution. No shooting, no blood, no guts, no gore.  The play is simple,
but you still must develop skill to play.  This version uses SVGAlib to
run on a graphics console.
```

- Sedaj bi radi videli, katere datoteke koules RPM namesti. Uporabite:

```
rpm -qpl koules-1.2-2.i386.rpm
```

Rezultat:

```
/usr/doc/koules
/usr/doc/koules/ANNOUNCE
/usr/doc/koules/BUGS
/usr/doc/koules/COMPILE.OS2
/usr/doc/koules/COPYING
/usr/doc/koules/Card
/usr/doc/koules/ChangeLog
/usr/doc/koules/INSTALLATION
/usr/doc/koules/Icon.xpm
/usr/doc/koules/Icon2.xpm
/usr/doc/koules/Koules.FAQ
/usr/doc/koules/Koules.xpm
/usr/doc/koules/README
/usr/doc/koules/TODO
/usr/games/koules
/usr/games/koules.svga
/usr/games/koules.tcl
/usr/man/man6/koules.svga.6
```

To je le nekaj primerov. Bolj kreativnih se lahko zlahka domislite, ko boste bolj domači v RPM.

6 Izgradnja RPM-jev

RPM-je je razmeroma enostavno zgraditi, še posebej če lahko dobite programsko opremo, ki jo želite zapakirati tako, da se bo lahko potem sama zgradila.

Osnovni potek izgradnje RPM-jev je naslednji:

- Prepričajte se, da je na vašem sistemu imenik `/etc/rpmrc` namestitveni imenik.
- Priskrbite si izvorno kodo za katero bi na vašem sistemu radi zgradili RPM.
- V vire vnesite vse popravke, ki so potrebni za pravilno izgradnjo.
- Za paket ustvarite specifikacijsko datoteko.
- Prepričajte se, da je vse na ustreznem mestu.

- S pomočjo RPM-ja zgradite paket.

Pri normalnem delovanju, RPM zgradi tako binarne, kot tudi izvorne pakete.

6.1 Datoteka rpmrc

Trenutno se nastavitve za RPM izvajajo samo preko datoteke `/etc/rpmrc`. Primer te datoteke:

```
require_vendor: 1
distribution: I roll my own!
require_distribution: 1
topdir: /usr/src/me
vendor: Mickiesoft
packager:  Mickeysoft Packaging Account <packages@mickiesoft.com>

signature: pgp
pgp_name: Mickeysoft Packaging Account
pgp_path: /home/packages/.pgp

tmppath: /usr/tmp
```

Vrstica `require_vendor` povzroči, da RPM poišče vrstico s podatki o prodajalcu. Ta podatek lahko izhaja iz `/etc/rpmrc` ali iz glave datoteke spec same. To izbiro izklopite tako, da postavite številko na 0. Enako velja za vrstici `require_distribution` in `require_group`.

Naslednja vrstica je vrstica s podatki o distribuciji. Definirate jo lahko na tem mestu, ali pa kasneje v glavi datoteke spec. Kadar gradite za določeno distribucijo, je dobro, da ta vrstica vsebuje pravilne podatke, čeprav ni obvezna. Vrstica s podatki o prodajalcu deluje na podoben način, definirana pa je lahko poljubno (npr. Andrejeva štacuna z rock glasbo in programsko opremo).

RPM podpira tudi izgradnjo paketov na različnih arhitekturah. Datoteka rpmrc lahko vsebuje spremenljivko „optflags“ potrebno za izgradnjo arhivov, ki med izgradnjo zahtevajo oznake za določene arhitekture. Za uporabo te spremenljivke si oglejte kasnejše razdelke.

Poleg tega makra, obstajajo še drugi. Uporabite lahko:

```
rpm --showrc
```

ter tako ugotovite vaše nastavitve in najdete vse oznake, ki so na voljo.

6.2 Datoteka Spec

Pričeli bomo z obravnavo datoteke spec. Ta datoteka je potrebna za izgradnjo paketov. Vsebuje opise programske opreme, skupaj z navodili kako jo zgraditi in seznam vseh binarnih zapisov, ki bodo nameščeni.

Svojo datoteko spec poimenujte v skladu s standardnim dogovorom. Imenovala naj bi se: *ime paketa* -vezaj- *številka različice* -vezaj- *številka izdaje* -pika- `spec`.

Oglejte si naslednjo majhno datoteko spec (`vim-3.0-1.spec`):

```
Summary: ejects ejectable media and controls auto ejection
Name: eject
Version: 1.4
Release: 3
Copyright: GPL
```

```
Group: Utilities/System
Source:
sunsite.unc.edu:/pub/Linux/utils/disk-management/eject-1.4.tar.gz
Patch: eject-1.4-make.patch
Patch1: eject-1.4-jaz.patch
%description
This program allows the user to eject media that is autoejecting like
CD-ROMs, Jaz and Zip drives, and floppy drives on SPARC machines.

%prep
%setup
%patch -p1
%patch1 -p1

%build
make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"

%install
install -s -m 755 -o 0 -g 0 eject /usr/bin/eject
install -m 644 -o 0 -g 0 eject.1 /usr/man/man1

%files
%doc README COPYING ChangeLog

/usr/bin/eject
/usr/man/man1/eject.1
```

6.3 Glava

Glava vsebuje nekaj standardnih polj, ki jim morate izpolniti. Vsebuje tudi nekaj opozoril. Polja je potrebno izpolniti na naslednji način:

- `Summary`: Enovrstičen opis paketa.
- `Name`: ime se mora ujemati z imenom datoteke rpm-ja, ki ga nameravate uporabiti.
- `Version`: ime različice se mora ujemati z imenom rpm-ja, ki ga nameravate uporabiti.
- `Release`: Številka izdaje paketa za isto verzijo (npr. če tvorimo paket in ugotovimo, da je rahlo okvarjen in ga je potrebno ponovno kreirati, bo naslednji paket imel številko izdaje 2).
- `Icon`: Ime datoteke z ikono, ki jo uporablja namestitveno orodje višje stopnje (kot npr. „glint“). To mora biti datoteka s končnico gif in mora biti v imeniku SOURCES.
- `Source`: Vrstica kaže na lokacijo prvotne izvorne datoteke HOME. Potrebovali jo boste, ko boste ponovno želeli zajeti izvorno kodo ali iskali novejše verzije. Opozorilo: pot do datoteke v tej vrstici SE MORA ujemati s potjo do datoteke, ki je na vašem sistemu (npr. ko boste zajeli izvorno datoteko, ji ne spremenite imena). S pomočjo naslednjih vrstic lahko definirate tudi več kot eno izvorno datoteko:

  ```
  Source0: blah-0.tar.gz
  Source1: blah-1.tar.gz
  Source2: fooblah.tar.gz
  ```

  Te vrstice bi šle v imenik SOURCES. (Struktura imenika je obravnavana v kasnejšem razdelku, „imeniška struktura izvora“.)

- Patch: To je mesto, kjer boste lahko našli popravek, če ga boste morali ponovno zajeti. Opozorilo: Ime datoteke se mora ujemati z imenom, ki ste ga uporabili, ko ste kreirali VAŠ popravek. Prav tako, kot lahko imate več virov, lahko imate tudi več popravkov.] Rezultat bo izgledal nekako tako:

  ```
  Patch0: blah-0.patch
  Patch1: blah-1.patch
  Patch2: fooblah.patch
  ```

  Te datoteke gredo v imenik SOURCES.

- Copyright: Ta vrstica pove komu pripadajo avtorske pravice. Uporabili bi naj nekaj takšnega kot GPL, BSD, MIT, public domain, distribute ali commercial.

- BuildRoot: Ta vrstica vam pri izgradnji in namestitvi omogoča, da imenik označite kot „korenski imenik“. To možnost lahko uporabite pri testiranju paketa, preden ga namestite na vaš računalnik.

- Group: Ta vrstica pove namestitvenim programom višje stopnje (kot je npr, „glint“) kam v hierarhični strukturi naj ga namesti. Trenutna drevesna struktura skupine izgleda nekako tako:

  ```
  Applications
      Communications
      Editors
          Emacs
      Engineering
      Spreadsheets
      Databases
      Graphics
      Networking
      Mail
      Math
      News
      Publishing
          TeX
  Base
      Kernel
  Utilities
      Archiving
      Console
      File
      System
      Terminal
      Text
  Daemons
  Documentation
  X11
      XFree86
          Servers
      Applications
          Graphics
          Networking
      Games
          Strategy
          Video
      Amusements
      Utilities
      Libraries
      Window Managers
  ```

```
Libraries
Networking
    Admin
    Daemons
    News
    Utilities
Development
    Debuggers
    Libraries
        Libc
    Languages
        Fortran
        Tcl
    Building
    Version Control
    Tools
Shells
Games
```

- `%description`: Ni ravno del glave, vendar mora biti definiran skupaj z glavo. Za en paket in/ali podpaket rabite en opisno označbo. To je več vrstično polje, ki bo moralo vsebovati razumljiv opis paketa.

6.4 Prep

Prep je drugi razdelek v datoteki spec. Uporablja se za pripravo virov na izgradnjo. Tu boste morali narediti vse, kar je potrebno za popravo in pripravo virov.

Pazite: Ta dva razdelka sta dejansko le prostora v katerem se izvrši zapis lupine. Za razpakiranje in popravo vaši virov, enostavno tvorite zapis sh in ga vstavite za označbo %prep. Za pomoč pri tem smo kreirali makre.

Prvi izmed teh makrov je makro %setup. V najpreprostejši obliki (brez stikal ukazne vrstice), preprosto razpakira vire in se pomakne (`cd`) v izvorno datoteko. V ukazu lahko uporabite tudi naslednja stikala:

- `-n ime` (name) nastavi ime zgrajenega imenika na izpisano ime. Privzeta vrednost je `$NAME-$VERSION`. Ostale možnosti vključujejo `$NAME`, `${NAME}${VERSION}` oz. karkoli glavna datoteka tar uporablja. (Ne pozabite, da spremenljivke „`$`“ niso prave spremenljivke, ki so na voljo v datoteki spec. Tu se uporabljajo samo kot primer. V vašem paketu morate uporabiti pravo ime in verzijo, ne pa spremenljivke.)
- `-c` tvori omenjeni imenik, se pomakne vanj in izvrši ukaz untar.
- `-b #` pred premaknitvijo (`cd`) v imenik razpakira Source# (to stikalo v kombinaciji s `c` nima nobenega smisla, torej je ne uporabljajte). Stikalo je uporabno samo v primeru različnih izvornih datotek.
- `-a #` najprej se pomakne v imenik in nato razpakira Source#.
- `-T` to stikalo prepiše privzeto opravilo za razpakiranje vira s programom tar in za razpakiranje glavne izvorne datoteke zahteva `-b 0` ali `-a 0`. Uporabite ga, kadar obstajajo sekundarni viri.
- `-D` imenika pred razpakiranjem ne zbriši. Stikalo je primerno samo v primeru, kadar imate več kot en nastavitveni makro. Uporabljalo naj bi se samo v naslednjih nastavitvenih makrih (nikoli pa ne v prvem makru).

Naslednji makro, ki je na voljo se imenuje makro %patch. Pomaga pri avtomatizaciji procesa uvajanja popravkov v vire. V njem lahko uporabite več stikal, ki so navedena spodaj:

- `#` bo uvedel `Patch#` kot popravljeno datoteko.

- `-p #` določi število imenikov, ki jih je potrebno izločiti iz ukaza `patch(1)`.
- `-P` privzeto opravilo je uporaba `Patch` (ali `Patch0`). Oznaka preprečuje privzeto opravilo in za razpakiranje glavne izvorne datoteke s programom tar zahteva vrednost 0. To stikalo je uporabno v drugem ali naslednjih makrih `%patch`, ki zahtevajo drugačno vrednost kot prvi makro.
- Namesto dejanskega ukaza `%patch # -P` lahko uporabite tudi `%patch#`.

To naj bi bili vsi makri, ki jih potrebujete. Ko jih boste pravilno sestavili, lahko opravite tudi druge nastavitve, ki jih potrebno opraviti s pomočjo zapisa tipa sh. Vse kar boste vnesli pred makrom `%build` (obravnavan v naslednjem razdelku) se izvede preko `sh`. V zgornjem primeru poiščite stvari, ki bi jih lahko opravili na tem mestu.

6.5 Izgradnja

V tem razdelku dejansko sploh ni makrov. Ko ste vir razpakirali, popravili in se premaknili v imenik, lahko v ta razdelek vnesete poljubne ukaze, ki jih pri izgradnji programske opreme potrebujete. To je še eno zaporedje ukazov, ki jih izvede `sh`, torej lahko vstavite vse dovoljene ukaze sh (vključno s komentarji). **Trenutni delovni imenik se v vsakem razdelku ponastavi na najvišjo stopnjo izvornega imenika.** Ne pozabite na to. Če bo potrebno, se lahko premaknete v podimenike z ukazom `cd`.

6.6 Namestitev

V tem razdelku sploh ni makrov. Vanj lahko vstavite poljubne ukaze, ki jih za namestitev potrebujete. Če ste si zagotovili namestitev v paketih, ki jih gradite, vnesite to sedaj. Drugače lahko za tvornjenje namestitve popravite makefile in izvršite namestitev, ali pa jo opravite ročno z ukazom sh. Vaš trenutni imenik lahko smatrate kot najvišjo stopnjo izvornega imenika.

6.7 Izbirne pred in po namestitvene/odstranjevalne skripte

Vstavite lahko zapise, ki se izvedejo pred in po namestitvi in odstranitvi binarnih paketov. Glavni razlog za to je, da lahko `ldconfig` zaženete po namestitvi ali odstranitvi paketov, ki vsebujejo deljene knjižnice. Makri za vsako skripto so v obliki:

- %pre je makro, ki izvrši pred-namestitvene skripte.
- %post je makro, ki izvrši po-namestitvene skripte.
- %preun je makro, ki izvrši pred-odstranitvene skripte.
- %postun je makro, ki izvrši po-odstranitvene skripte.

Vsebina teh razdelkov mora biti v obliki zapisa `sh`, čeprav *ne* potrebujete `#!/bin/sh`.

6.8 Datoteke

To je razdelek, kjer morate navesti datoteke za binarni paket. RPM ne ve, katere binarne datoteke se namestijo kot rezultat ukaza `make install`. Za to *NE* obstaja noben način. Nekateri so predlagali, da je potrebno pred in po namestitvi paketa izvajati iskanje. Na večuporabniškem sistemu, to ni sprejemljivo, ker se lahko med procesom izgradnje paketov tvorijo druge datoteke, ki s paketom nimajo nič skupnega.

Na voljo je nekaj makrov, ki opravljajo tudi posebna opravila. Navedeni in opisani so spodaj:

- `%doc` se uporablja za označevanje dokumentacije v izvornem paketu, ki ga želite namestiti v binarni obliki. Spis se bodo namestili v `/usr/doc//$NAME-$VERSION-$RELEASE`. S tem makrom lahko v opravilni vrstici prelistate tudi več spisov hkrati, ali pa ločeno z uporabo makra za vsakega posebej.
- `%config` se uporablja za označevanje nastavitvenih datotek v paketu. Sem spadajo datoteke kot so `sendmail.cf`, `passwd`, itd. Če paket, ki vsebuje nastavitvene datoteke kasneje odstranite, se bodo odstranile vse nespremenjene datoteke, vse spremenjene datoteke pa bodo dobile prvotno ime s pripono `.rpmsave`. Z makrom lahko prelistate tudi več datotek.
- `%dir` označuje samostojen imenik v seznamu datotek, da je vključen, kot bi spadal k paketu. Če prelistate ime imenika *BREZ UPORABE* makra `%dir`, bo glede na privzeto vrednost v seznam datotek vključeno *VSE* kar ta imenik vsebuje in kasneje nameščeno kot del paketa.
- `%files f` <`ime datoteke`> vam omogoča listanje datotek v poljubni datoteki znotraj gradbenega imenika vira. To je koristno v primerih, ko imate paket, ki lahko zgradi svoj lasten seznam datotek. Ta seznam tukaj preprosto vključite in vam ne bo treba podajati seznama datotek.

Največjo nevarnost v seznamu datotek predstavlja listanje imenikov. Če prelistate `/usr/bin` po nesreči, bo vaš binarni paket vseboval vse datoteke v `/usr/bin` na vašem sistemu.

6.9 Izgradnja

6.9.1 Struktura imenika vira

Najprej potrebujete pravilno oblikovano gradbeno strukturo. To lahko opravite z uporabo datoteke `/etc/rpmrc`. Večina ljudi uporablja kar `/usr/src`.

Mogoče boste morali za tvorjenje gradbene strukture ustvarili naslednje imenike:

- `BUILD` je imenik, kjer RPM izvrši celotno izgradnjo. Čeprav ni potrebno, da testne izgradnje vršite v točno določenem imeniku, bo RPM vršil izgradnjo vedno v tem imeniku.
- `SOURCES` je imenik, kamor bi morali vstaviti originalne izvorne datoteke tar in popravke. RPM bo, glede na privzeto nastavitev, iskal v tem imeniku.
- `SPECS` je imenik, kjer naj bi bile vse datoteke spec.
- `RPMS` je imenik, kamor bo po izgradnji RPM shranil binarne RPM-je.
- `SRPMS` je imenik, kjer bodo shranjeni vsi izvorni RPM-ji.

6.9.2 Testna izgradnja

Verjetno boste najprej poskusili dobiti izvorno kodo, da bi izgradili čisto brez uporabe RPM. Da bi to storili, najprej razpakirajte vir in spremenite ime imenika v $NAME.orig. Potem ponovno razpakirajte vir. Pri izgradnji uporabite ta vir. Premaknite se v izvorni imenik in za izgradnjo sledite navodilom. Če boste morali stvari urejati, boste potrebovali popravek. Ko se bo izgradil, počistite izvorni imenik. Prepričajte se, da ste odstranili vse datoteke, ki jih je tvorila nastavitvena skripta `configure`. Potem se iz izvornega imenika pomaknite v njegov predhodni imenik in izvedite nekaj takšnega:

```
diff uNr dirname.orig dirname > ../SOURCES/dirname-linux.patch
```

S tem boste ustvarili popravek, ki ga lahko uporabite v datoteki spec. Ne pozabite, da "linux", ki ga vidite v imenu popravka, služi samo identifikaciji. Mogoče boste za opis namena popravka uporabili tudi kaj bolj opisnega, kot npr. čonfig" ali "bugs". Dobro je, da pred uporabo datoteke za popravek, ki jo ustvarite, le to pred tem pregledate in se prepričate, da ste vključili samo tiste binarne datoteke, ki jih želite.

6.9.3 Tvorjenje seznama datotek

Sedaj, ko imate vir, ki bo izgradil paket, in ko veste, kako to narediti, ga izgradite in namestite. Oglejte si izhod namestitvenega niza in z njegovo pomočjo izgradite seznam datotek, ki ga boste uporabili v datoteki spec. Ponavadi gradimo datoteko spec vzporedno z vsemi temi koraki. Tvorite lahko začetno datoteko in vstavite enostavnejše dele, potem po korakih vstavite še ostale.

6.9.4 Izgradnja paketov z RPM

Ko imate datoteko spec, ste pripravljeni na izgradnjo svojega paketa. Najuporabnejši način za to opravilo je uporaba ukaz v naslednji obliki:

```
rpm ba foobar-1.0.spec
```

Skupaj s stikalom `-b` lahko koristno uporabite tudi druga stikala:

- `p` pomeni: zaženi samo razdelek prep datoteke spec.
- `l` je namenjen preverjanju seznama in opravi nekaj testov na `%files`.
- `c` tvori prep in prevod. To je primeren način, kadar niste prepričani, ali se bo vaš izvor sploh zgradil. Zdi se neuporaben, ker se verjetno želite igrati z virom tako dolgo, dokler se ne zgradi in potem uporabiti RPM. Ko pa se boste privadili uporabe RPM, boste spoznali, da vam bo v nekaterih primerih v pomoč.
- `i` napravi prep, prevede in namesti.
- `b` napravi prep, prevede, namesti in izgradi samo binarni paket.
- `a` zgradi vse (tako izvorne, kot tudi binarne pakete).

Za stikalo b obstaja tudi nekaj določil:

- `-short-circuit` bo skočil neposredno na določeno stopnjo (uporablja se lahko samo v kombinaciji s `c` in `i`).
- `-clean` odstrani gradbeno strukturo, ko je gotova.
- `-keep-temps` ohranijo se vse začasne datoteke in skripte, ki so bili ustvarjene v `/tmp`. Z uporabo stikala `-v` lahko vidite, katere datoteke so se ustvarile v `/tmp`.
- –test ne izvede dejanskih stopenj, vendar izvede keep-temp.

6.10 Testiranje

Ko imate izvorni in binarni rpm za svoj paket, ju je potrebno testirati. Najenostavnejše in najbolj učinkovito je, da ga preizkusite na popolnoma drugem računalniku. Nenazadnje ste izvršili veliko ukazov `make install` na vašem računalniku, torej mora biti dokaj dobro nameščen.

Na paketu lahko uporabite `rpm -u imepaketa` in ga tako testirate. To pa lahko daje zavajajoče rezultate, ker ste pri izgradnji paketa uporabili `make install`. Če ste nekaj s seznama datotek izpustili, se to ne bo odstranilo. Potem boste binaren paket ponovno namestili in vaš sistem bo spet popoln, vaš rpm pa še vedno ne bo. Ne pozabite, da bo zaradi tega ker ste za kreiranje paketa uporabili `rpm -ba`, večina ljudi pri namestitvi uporabila samo `rpm -i`. V razdelkih izgradnje in namestitve ne storite ničesar, kar bi bilo potrebno storiti kadar se binarne datoteke nameščajo samodejno.

6.11 Kaj narediti z novimi RPM-ji

Ko ste naredili svoj RPM nečesa (predpostavljamo, da to nekaj ni bil že RPM-jan), lahko svoje delo razdelite ostalim (predpostavljamo tudi, da ste RPM-jali nekaj, kar se lahko prosto razširja). To lahko storite tako, da paket naložite na `<ftp://ftp.redhat.com/>`.

6.12 Kaj sedaj?

Oglejte si zgornja razdelka o testiranju in kaj narediti z novimi RPM-ji. Želimo, da so na voljo vsi RPM-ji, ki jih dobimo in, da so dobro narejeni. Vzemite si čas in jih dobro preskusite, potem pa jih naložite v dobro vsem. *Prosim*, prav tako se prepričajte, da nalagate samo *programsko opremo, ki je prosto dostopna*. Komercialne programske oprema *ne* nalagajte, če avtorske pravice tega ne dopuščajo. Sem spadajo programska oprema Netscape, `ssh`, `pgp`, itd.

7 Več-arhitekturna izgradnja RPM-jev

RPM se lahko sedaj uporablja za izgradnjo paketov za procesorje Intel i386, Digital Alpha, ki uporabljajo Linux in Sparc. Obstajajo poročila, da deluje tudi na SGI in HP delovnih postajah. Obstajajo številne lastnosti, ki poenostavljajo izgradnjo paketov na vseh platformah. Prva izmed njih je direktiva "optflag" v `/etc/rpmrc`. Uporablja se lahko za postavitev oznak, ki se uporabljajo pri izgradnji programske opreme za vrednosti, ki so specifične za določeno arhitekturo. Naslednja lastnost so makri "arch" v datoteki spec. Uporabljate jih lahko v različne namene, odvisno od arhitekture na kateri gradite. Naslednja lastnost je direktiva "Exclude" v glavi.

7.1 Primer datoteke spec

Sledi del datoteke spec za paket "fileutils". Zgrajena je tako, da se lahko zgradi na računalnikih s procesorji Alpha in Intel.

```
Summary: GNU File Utilities
Name: fileutils
Version: 3.16
Release: 1
Copyright: GPL
Group: Utilities/File
Source0: prep.ai.mit.edu:/pub/gnu/fileutils-3.16.tar.gz
Source1: DIR_COLORS
Patch: fileutils-3.16-mktime.patch

%description
These are the GNU file management utilities.  It includes programs
to copy, move, list, etc, files.

The ls program in this package now incorporates color ls!

%prep
%setup

%ifarch alpha
%patch -p1
autoconf
%endif
%build
```

```
configure --prefix=/usr --exec-prefix=/
make CFLAGS="$RPM_OPT_FLAGS" LDFLAGS=-s

%install
rm -f /usr/info/fileutils*
make install
gzip -9nf /usr/info/fileutils*

.
.
.
```

7.2 Optflags

V tem primeru boste videli, kako se direktiva "optflag" uporablja iz `/etc/rpmrc`. Glede na arhitekturo na kateri gradite, se ustrezna vrednost posreduje `RPM_OPT_FLAGS`. Da bi lahko namesto navadne direktive, ki jo uporabljate (kot je npr. `-m486` ali `-O2`), uporabljali to spremenljivko, boste morali popraviti Makefile za vaš paket. Boljši občutek za to, kaj je potrebno storiti, lahko dobite tako, da namestite ta izvorni paket, ga potem razpakirate in preučite Makefile. Potem si oglejte popravek za Makefile in videli boste, kakšne spremembe je potrebno opraviti.

7.3 Makri

Makro `%ifarch` je pri vsem tem zelo pomemben. Največkrat boste morali narediti popravek ali dva, ki je značilen samo za določeno arhitekturo. V tem primeru vam RPM dopušča, da napravite popravek samo za to arhitekturo.

V zgornjem primeru ima fileutils popravke za 64 bitne računalnike. V tem trenutku se ta popravek uporablja samo za procesorje Alpha. Okrog popravka za 64 bitne procesorje dodajmo makro `%ifarch`:

```
%ifarch axp
%patch1 -p1
%endif
```

S tem boste zagotovili, da se bo popravek uporabljal samo na arhitekturah s procesorji Alpha.

7.4 Izločevanje arhitektur iz paketa

Da boste lahko vzdrževali izvorne RPM-je za vse platforme v enem imeniku, smo vgradili sposobnost za "izločevanje" paketov, ki bi se zgradili za določene arhitekture in sicer tako, da boste še vedno lahko uporabljali ukaze, kot je

```
rpm --rebuild /usr/src/SRPMS/*.rpm
```

in pri tem izgradite ustrezen paket. Če aplikacije še niste dali na določeno platformo, je vse kar morate storiti, da dodate podobno vrstico:

```
ExcludeArch: axp
```

v glavo datoteke spec izvornega paketa. Potem ponovno izgradite paket na platformi, na kateri je prvotno izgrajena. Dobili boste izvorni paket, ki deluje s procesorjem Intel, lahko ga pa enostavno prenesete na računalnik s procesorjem Alpha.

7.5 Zaključek

Uporaba RPM-ja za izgradnjo več-arhitekturnih paketov je po navadi bolj enostavno, kot pa dobiti pakete za samo izgradnjo. Več kot je paketov, lažje je to opraviti. Kot vedno, je najboljša pomoč, ko pri izgradnji RPM-jev obtičite ta, da pogledate v podoben izvorni paket.

8 O avtorskih pravicah