Ελληνικό RPM HOWTO

Donnie Barnes, djb@redhat.com

8 Απριλίου 1997, έκδοση v2.07

Contents

1 Εισαγωγή

Το RPM είναι είναι ακρωνύμιο του **R**ed Hat **P**ackage **M**anager. Παρόλο που περιέχει το Red Hat στο όνομα του, η πρόθεση είναι να αποτελέσει ένα ανοικτό σύστημα πακεταρίσματος, διαθέσιμο προς χρήση από καθέναν. Επιτρέπει στους χρήστες να παίρνουν τον πηγαίο κώδικα ενός νέου λογισμικού και να τον πακετάρουν σε πηγαία και εκτελέσιμη μορφή, ώστε να μπορούν να εγκαθίστανται και να παρακολουθούνται εύκολα τα εκτελέσιμα πακέτα, και ο πηγαίος κώδικας να μπορεί να χτίζεται με απλό τρόπο.

Διατηρεί επίσης μια βάση δεδομένων όλων των πακέτων και των αρχείων τους, που μπορεί να χρησιμοποιηθεί για την επαλήθευση της ύπαρξης και ακεραιότητας των πακέτων καθώς και για την απάντηση σε ερωτήσεις σχετικά με αρχεία ή πακέτα.

Η εταιρεία Red Hat Software ενθαρρύνει άλλους προμηθευτές διανομών να διαθέσουν χρόνο στο να κατανοήσουν το RPM και να το χρησιμοποιήσουν για τις δικές τους διανομές. Το RPM είναι πολύ ευέλικτο και εύχρηστο, παρόλο που παρέχει την βάση για ένα πολύ ευρύ σύστημα. Είναι επίσης τελείως ανοικτό και πλήρως διαθέσιμο, και θα εκτιμούσαμε οποιεσδήποτε αναφορές σφαλμάτων και διορθώσεις. Εκχωρείται η άδεια δωρεάν χρήσης και διανομής του RPM κάτω από τους όρους της GPL.

Μια πληρέστερη τεκμηρίωση για το RPM είναι διαθέσιμη στο βιβλίο του Ed Bailey, *Maximum RPM*. Το βιβλίο είναι διαθέσιμο για να το κατεβάσετε ή να το αγοράσετε στη διεύθυνση *www.redhat.com* `<http://www.redhat.com>`.

2 Επισκόπηση

Πρώτα απ'όλα, επιτρέψτε μου να περιγράψω τα βασικά της φιλοσοφίας που κρύβεται πίσω από το RPM. Ένας σχεδιαστικός στόχος ήταν να επιτρέψει την χρήση "πρωταρχικού (βασικού)" κώδικα. Με το RPP (ο προηγούμενος μας διαχειριστής πακέτων, από τον οποίον δε χρησιμοποιήθηκε *τίποτα* στη δημιουργία του RPM), τα πακέτα πηγαίου κώδικα ήταν "πειραγμένοι" αρχικοί πηγαίοι κώδικες που μετά χτίσαμε. Θεωρητικά, κάποιος θα μπορούσε να εγκαταστήσει ένα πηγαίο RPP και μετά να κάνει`make` χωρίς προβλήματα. Αλλά ο πηγαίος κώδικας δεν ήταν ο αρχικός πρωτότυπος, και δεν υπήρχε καμία αναφορά στο ποιες αλλαγές χρειάστηκε να κάνουμε για να τα καταφέρουμε να χτίζεται επιτυχώς. Θα έπρεπε να κατεβάσετε τον αρχικό πρωτότυπο κώδικα ξεχωριστά. Με το ROM, έχετε τον αρχικό κώδικα μαζί με ένα μπάλωμα (patch) το οποίο χρησιμοποιήσαμε για να κάνουμε τη μεταγλώττιση. Αυτό το θεωρούμε σαν ένα μεγάλο πλεονέκτημα. Γιατί; Για πολλούς λόγους. Πρώτα - πρώτα, αν βγει μια νέα έκδοση κάποιου προγράμματος, δε χρειάζεται να αρχίσετε αναγκαστικά από την αρχή για να το μεταγλωττίσετε στο RHL. Μπορείτε να κοιτάξετε στο μπάλωμα (patch) για το τι *ίσως χρειαστεί* να κάνετε. Με αυτόν τον τρόπο, όλες οι προκαθορισμένες ρυθμίσεις της μεταγλώττισης είναι εύκολα κατανοητές.

Το RPM σχεδιάστηκε για να προσφέρει ισχυρές δυνατότητες ερώτησης. Μπορείτε να ψάξετε μέσα σε ολόκληρη τη βάση σας για πακέτα ή για ολόκληρα αρχεία. Μπορείτε επίσης να βρείτε εύκολα σε ποιο πακέτο ανήκει ένα συγκεκριμένο αρχείο και από ποιό προήλθε. Τα ίδια τα RPM αρχεία είναι συμπιεσμένα αρχεία, αλλά μπορείτε να κάνετε εύκολα και *σύντομα* μια ερώτηση σε ξεχωριστά πακέτα χάρη σε μια επικεφαλίδα δυαδικής μορφής, που έχει προστεθεί στο πακέτο και περιέχει οτιδήποτε χρειάζεται να γνωρίζετε. Αυτό συντελεί σε *γρήγορη* ερώτηση.

Άλλο ένα ισχυρό χαρακτηριστικό είναι η ικανότητα να επαληθεύσετε πακέτα. Αν ανησυχείτε ότι σβήσατε ένα σημαντικό αρχείο για κάποιο πακέτο, απλά επαληθεύστε το. Θα ενημερωθείτε για τυχόν ανωμαλίες. Σε αυτή την περίπτωση, μπορείτε να επανεγκαταστήσετε το πακέτο αν είναι απαραίτητο. Ο,τι αρχεία ρυθμίσεων είχατε θα παραμείνουν ακέραια.

Θα θέλαμε να ευχαριστήσουμε τα παιδιά της διανομής BOGUS μια και πολλές από τις ιδέες τους χρησιμοποιήθηκαν στο RPM. Αν και το RPM γράφτηκε αποκλειστικά από τη Red Hat Software, η λειτουργία του είναι βασισμένη στον κώδικα που γράφτηκε για τη διανομή BOGUS (PM και PMS).

3 Γενικές Πληροφορίες

3.1 Αποκτώντας το RPM

Ο καλύτερος τρόπος να αποκτήσετε το RPM είναι να στήσετε το Red Hat Linux. Αν δε θέλετε να το κάνετε αυτό, μπορείτε και πάλι να πάρετε και να χρησιμοποιήσετε το RPM. Θα το βρείτε στο *ftp.redhat.com* `<ftp://ftp.redhat.com/pub/redhat/code/rpm>`.

3.2 Απαιτήσεις του RPM

Η κύρια απαίτηση για να τρέξετε το RPM είναι να έχετε το cpio 2.4.2 ή μεγαλύτερο. Αν και το σύστημα προορίζεται για χρήση με Linux, μπορεί να μεταφερθεί και σε άλλα συστήματα Unix. Συγκεκριμένα, έχει μεταγλωττιστεί για SunOS, Solaris, AIX, Irix, AmigaOS και άλλα. Σας προειδοποιούμε ότι τα δυαδικά-εκτελέσιμα πακέτα που δημιουργείτε σε ένα σύστημα Unix δε θα είναι συμβατά σε άλλο σύστημα.

Αυτές είναι οι ελάχιστες απαιτήσεις για να εγκαταστήσετε RPMs. Για να χτίσετε RPMs από τον κώδικα, χρειάζεστε ότι θα θέλατε κανονικά για να χτίσετε ένα πακέτο, όπως τα `gcc`, `make`, κλπ.

4 Χρησιμοποιώντας το RPM

Στην απλούστερη μορφή του, το RPM χρησιμοποιείται για να εγκαταστήστε πακέτα:

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

Η επόμενη απλούστατη εντολή απεγκαθιστά ένα πακέτο:

```
rpm -e foobar
```

Μία από τις πιο σύνθετες αλλά και *πολύ* χρήσιμες εντολές σας επιτρέπει να εγκαταστήσετε πακέτα μέσω FTP. Αν είστε συνδεδεμένοι στο δίκτυο και θέλετε να εγκαταστήσετε ένα νέο πακέτο, το μόνο που χρειάζεται είναι να προσδιορίσετε το αρχείο δίνοντας το κατάλληλο URL, όπως :

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

Παρακαλώ σημειώστε ότι το RPM θα κάνει την ερώτηση και/ή θα εγκαταστήσει το πακέτο μέσω FTP.

Ενώ αυτές είναι απλές εντολές, το rpm μπορεί να χρησιμοποιηθεί με πολλούς τρόπους όπως θα δείτε από το μύνημα `Usage` :

```
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}
```

Μπορείτε να βρείτε περισσότερες λεπτομέρειες για τις επιλογές του RPM στη σελίδα βοηθείας (man page).

5 Και τι μπορώ να κάνω *πραγματικά* με το RPM;

Το RPM είναι πολύ χρήσιμο εργαλείο, και όπως θα δείτε, έχει πολλές επιλογές. Ο καλύτερος τρόπος για να βγάλετε μια άκρη με αυτές είναι να δείτε μερικά παραδείγματα. Δείξαμε παραπάνω την επιλογή εγκατάστασης/απεγκατάστασης και τώρα ακολουθούν μερικά ακόμη παραδείγματα :

- Ας πούμε ότι διαγράψατε μερικά αρχεία κατά λάθος, αλλά δεν είστε σίγουροι για το ποια σβήσατε. Αν θέλετε να επαληθεύσετε το σύστημά σας και να δείτε τι λείπει, θα γράψετε :

```
rpm -Va
```

- Ας πούμε ότι συναντάτε κάποιο αρχείο που δεν αναγνωρίζετε. Για να βρείτε σε ποιο πακέτο ανήκει, θα γράψετε :

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

Το αποτέλεσμα θα είναι:

```
xjewel-1.6-1
```

- Βρίσκετε ένα νέο RPM με το όνομα koules, αλλά δεν ξέρετε τι είναι αυτό. Για να δείτε κάποιες πληροφορίες σχετικά με το πακέτο, θα γράψετε :

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

Το αποτέλεσμα θα είναι:

```
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.
```

- Τώρα θέλετε να δείτε ποια αρχεία εγκαθιστά το RPM koules. Θα γράψετε:

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

Το αποτέλεσμα είναι:

```
/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
```

Αυτά ήταν μερικά παραδείγματα. Μπορείτε να επινοήσετε πιο δημιουργικές χρήσεις του RPM όταν εξοικειωθείτε με αυτό.

6 Χτίζοντας RPMs

Είναι αρκετά εύκολο να χτίσετε RPMs, ειδικότερα αν μπορείτε να καταφέρετε να χτίσετε από μόνο του το λογισμικό που σας ενδιαφέρει να πακετάρετε.

Η βασική διαδικασία για να χτίσετε ένα RPM είναι η ακόλουθη:

- Ελέγξτε ότι το `/etc/rpmrc` έχει εγκατασταθεί στο σύστημά σας.
- Κάντε τον πηγαίο κώδικα για τον οποίο θα χτίστε το RPM να περνάει από μεταγλώττιση στο σύστημά σας.
- Φτιάξτε ένα μπάλωμα (patch) για κάθε αλλαγή που χρειαστήκατε να κάνετε ώστε να μεταγλωττιστεί σωστά
- Φτιάξτε ένα αρχείο προδιαγραφών (spec file) για το πακέτο.
- Βεβαιωθείτε πως όλα είναι στη σωστή τους θέση.
- Χτίστε το πακέτο χρησιμοποιώντας το RPM.

Υπό φυσιολογικές συνθήκες, το RPM δημιουργεί τόσο το πακέτο του εκτελέσιμου όσο και του πηγαίου κώδικα.

6.1 Το αρχείο rpmrc

Για την ώρα, η μόνη ρύθμιση του RPM διατίθεται μέσω του αρχείου `/etc/rpmrc`. Ακολουθεί ένα παράδειγμα:

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

optflags: i386 -O2 -m486 -fno-strength-reduce
optflags: alpha -O2
optflags: sparc -O2

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

tmppath: /usr/tmp
```

Η γραμμή `require_vendor` κάνει το RPM να ζητά να βρει μια γραμμή πωλητή (vendor). Αυτή μπορεί να προέλθει από το αρχείο `/etc/rpmrc` ή από την επικεφαλίδα του ίδιου του αρχείου προδιαγραφών (spec file). Για να το απενεργοποιήσετε, αλλάξτε τον αριθμό σε 0. Το ίδιο ισχύει για τις γραμμές `require_distribution` και `require_group`.

Η επόμενη γραμμή είναι σχετική με τη διανομή `distribution`. Μπορείτε να τη δηλώσετε εδώ ή αργότερα στην επικεφαλίδα του αρχείου προδιαγραφών (spec file). Όταν χτίζετε το πακέτο για κάποια συγκεκριμένη διανομή, είναι καλή ιδέα να βεβαιωθείτε ότι αυτή η γραμμή είναι σωστή, ακόμη κι αν δεν απαιτείται. Η γραμμή `vendor` δουλεύει αντίστοιχα, αλλά μπορεί να γράφει οτιδήποτε (π.χ. Joe's Software and Rock Music Emporium).

Το RPM έχει υποστήριξη για να χτίσετε πακέτα σε πολλαπλές αρχιτεκτονικές. Το αρχείο `rpmrc` μπορεί να κρατά μια μεταβλητή "optflags" για να χτίσετε πράγματα που απαιτούν συγκεκριμένα flags ανάλογα με την αρχιτεκτονική κατά το χτίσιμό τους. Δείτε τα παρακάτω τμήματα για το πώς να χρησιμοποιήσετε αυτή τη μεταβλητή.

Υπάρχουν και άλλα πολλά επιπλέον macros που μπορείτε να χρησιμοποιήσετε για να δείτε τα tags και τα διαθέσιμα flags.

```
rpm --showrc
```

6.2 Το αρχείο προδιαγραφών (Spec File)

Θα ξεκινήσουμε με συζήτηση για το αρχείο προδιαγραφών. Τα αρχεία προδιαγραφών απαιτούνται για να χτίσετε ένα πακέτο. Το αρχείο προδιαγραφών είναι μια περιγραφή του λογισμικού μαζί με εντολές και οδηγίες για το πώς θα στηθούν και μία λίστα για όλα τα εκτελέσιμα που εγκαταστάθηκαν.

Καλό είναι να ονομάσετε το αρχείο προδιαγραφών σύμφωνα με τυπικές συμβάσεις. Θα πρέπει να είναι το πακέτο όνομα-παύλα-αριθμός έκδοσης (version number)-παύλα- αριθμός σειράς (release number)-τελεία-spec.

Ακολουθεί ένα μικρό παράδειγμα αρχείου προδιαγραφών (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
\textgreek{Aut'o} \textgreek{to} \textgreek{pr'ogramma} \textgreek{epitr'epei} \textgreek{sto} \textgreek{qr'hsth} \textgreek{na} \textgreek{bg'azei} (eject) \textgreek{ta} \textgreek{m'esa} \textgreek{pou} \textgreek{uposthr'izoun} \textgreek{aut'omath} \textgreek{exagwg'h} \textgreek{'opwc} CD-ROMs,Jaz \textgreek{kai} Zip drives, \textgreek{kai} \textgreek{odhgo'uc} \textgreek{disk'etac} \textgreek{se} \textgreek{mhqan'ec} SPARC.

%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 Η επικεφαλίδα

Η επικεφαλίδα έχει μερικά σταθερά πεδία που πρέπει να συμπληρωθούν. Δίνονται επίσης και μερικές υποδείξεις. Τα πεδία πρέπει να συμπληρωθούν όπως φαίνεται παρακάτω:

- **Summary:** Είναι η περιγραφή του πακέτου σε μια γραμμή.

- `Name:` Είναι το όνομα του rpm αρχείου που σκοπεύετε να χρησιμοποιήσετε.
- `Version:` Είναι η έκδοση (version) του rpm αρχείου που σκοπεύετε να χρησιμοποιήσετε.
- `Release:` Είναι η σειρά (release) του πακέτου για την παραπάνω έκδοση (π.χ. αν φτιάξουμε ένα πακέτο και το βρούμε λίγο προβληματικό και πρέπει να το ξαναφτιάξουμε, το επόμενο πακέτο θα είναι σειρά 2).
- `Icon:` Είναι το όνομα του αρχείου του εικονιδίου που θα χρησιμοποιηθεί με άλλα εργαλεία εγκατάστασης υψηλού επιπέδου (όπως το "glint" της Red Hat). Πρέπει να είναι gif και να βρίσκεται στον κατάλογο SOURCES.
- `Source:` Αυτή η γραμμή δείχνει στην τοποθεσία HOME του αρχικού αρχείου πηγαίου κώδικα. Χρησιμοποιείται στην περίπτωση που θέλετε να πάρετε τον κώδικα ξανά ή να ελέγξετε για νεώτερες εκδόσεις. Σημείωση: Το όνομα αρχείου ΠΡΕΠΕΙ να ταιριάζει με το όνομα του αρχείου που έχετε στο σύστημά σας (π.χ. μην κατεβάσετε το πηγαίο αρχείο και αλλάξετε μετά το όνομά του). Μπορείτε επίσης να καθορίσετε περισσότερα του ενός αρχεία πηγαίου κώδικα χρησιμοποιώντας γραμμές όπως :

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

  Αυτά τα αρχεία πηγαίνουν στον κατάλογο `SOURCES` . (Η δομή του καταλόγου θα συζητηθεί αργότερα σε επόμενο τμήμα, "Ο δενδροειδής κατάλογος του πηγαίου κώδικα").

- `Patch:` Είναι το μέρος που θα βρείτε το μπάλωμα (patch) αν χρειαστεί να το κατεβάσετε ξανά. Σημείωση: Το όνομα αρχείου εδώ πρέπει να ταιριάζει με αυτό που χρησιμοποιείτε όταν φτιάχνετε το δικό ΣΑΣ patch. Μπορεί επίσης να θέλετε να σημειώσετε ότι μπορεί να έχετε πολλαπλά patch αρχεία όσο μπορεί να έχετε πολλαπλά πηγαία αρχεία. Θα έχετε κάτι σαν το παρακάτω:

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

  Αυτά τα αρχεία θα πάνε στον κατάλογο `SOURCES` .

- `Copyright:` Αυτή η γραμμή αναφέρει για τα δικαιώματα του πακέτου. Θα πρέπει να χρησιμοποιείτε κάτι σαν GPL, BSD, MIT, public domain, distributable, ή commercial.
- `BuildRoot:` Αυτή η γραμμή σας επιτρέπει να καθορίσετε έναν κατάλογο σαν "root" για να χτίσετε και να εγκαταστήσετε ένα πακέτο. Μπορείτε να το χρησιμοποιήσετε για να τεστάρετε το πακέτο σας πριν να το εγκαταστήσετε στο μηχάνημά σας.
- `Group:` Αυτή η γραμμή χρησιμοποιείται για να πει σε προγράμματα εγκατάστασης υψηλού επιπέδου (όπως το "glint" της RedHat) πού να βάλει αυτό το πρόγραμμα στη δομή της ιεραρχίας. Το δένδρο του γκρουπ μοιάζει ως εξής :

```
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` Δεν είναι πραγματικά ένα τμήμα της επικεφαλίδας, αλλά θα πρέπει να περιγραφεί με το υπόλοιπο της επικεφαλίδας. Χρειάζεστε ένα tag περιγραφής ανά πακέτο και/ή υποπακέτο. Αυτό είναι ένα πεδίο πολλαπλών-γραμμών που πρέπει να χρησιμοποιηθεί για να δώσει μια κατανοητή περιγραφή του πακέτου.

6.4 Prep

Αυτό είναι το δεύτερο τμήμα στο αρχείο προδιαγραφών. Χρησιμοποιείται για να ετοιμάσετε τον κώδικα για χτίσιμο. Εδώ πρέπει να κάνετε οτιδήποτε χρειαστεί για να κάνετε patch τους κώδικες και να τους εγκαταστήσετε όπως πρέπει για να γράψετε `make`.

Σημειώστε το εξής: Κάθε ένα από αυτά τα τμήματα είναι πραγματικά χώρος για να εκτελέσετε σκριπτάκια φλοιού (shell scripts). Μπορείτε απλά να φτιάξετε ένα `sh` script και να το βάλετε μετά το `%prep` tag για να ξεπακετάρετε και να κάνετε patch τον κώδικά σας. Παρόλαυτά, έχουμε φτιάξει macros για να βοηθήσουμε ς'αυτό.

Το πρώτο απ'αυτά τα macros είναι το `%setup` macro. Στην απλούστερή του μορφή (χωρίς επιλογές από τη γραμμή εντολών), απλά ξεπακετάρει τον κώδικα και μπαίνει στον κατάλογο με τον κώδικα. Παίρνει επίσης τις παρακάτω επιλογές:

- `-n name` θέτει το όνομα του καταλόγου που θα χρησιμοποιηθεί για το χτίσιμο στο όνομα που ακολουθεί `name`. Το προκαθορισμένο όνομα είναι `$ONOMA-$ΕΚΔΟΣΗ`. Άλλες δυνατότητες περιλαμβάνουν το `$ONOMA`, `${ONOMA}${ΕΚΔΟΣΗ}`, ή ότι χρησιμοποιεί το κύριο tar αρχείο. (Σημειώστε παρακαλώ ότι οι μεταβλητές "$" *δεν* είναι πραγματικές μεταβλητές που διατίθενται μέσα στο αρχείο προδιαγραφών. Χρησιμοποιούνται εδώ αντί για ένα όνομα-παράδειγμα. Εσείς πρέπει να χρησιμοποιήσετε το πραγματικό όνομα και την έκδοση στο πακέτο σας, όχι μια μεταβλητή).
- `-c` θα δημιουργήσει και θα μπει στον ονομαζόμενο κατάλογο *πριν* να κάνει untar.
- `-b #` θα κάνει untar τον Κώδικα# *πριν* να μπει στον κατάλογο (και δε βγάζει νόημα με την επιλογή `-c` γι'αυτό μην το κάνετε). Αυτό είναι χρήσιμο όταν έχετε πολλαπλά αρχεία με κώδικα.
- `-a #` θα κάνει untar τον Κώδικα# *αφού* μπει στον κατάλογο.
- `-T` Αυτή η επιλογή αγνοεί την προκαθορισμένη επιλογή που κάνει untar τον Κώδικα και απαιτεί την επιλογή `-b 0` ή `-a 0` για να κάνετε untar το κύριο αρχείο κώδικα. Θα το χρειαστείτε αυτό όταν υπάρχουν δευτερεύουσες πηγές.
- `-D` *ΜΗ* διαγράφετε τον κατάλογο πριν να ξεπακετάρετε. Αυτό είναι χρήσιμο μόνο όπου έχετε περισσότερα του ενός macro εγκατάστασης. Θα πρέπει να χρησιμοποιείται *μόνο* σε macros εγκατάστασης *μετά* το πρώτο (αλλά ποτέ στο πρώτο).

Το επόμενο από τα διαθέσιμα macros είναι το `%patch` macro. Αυτό το macro βοηθά στην αυτοματοποίηση της διαδικασίας της εφαρμογής μπαλωμάτων (patches) στους κώδικες. Παίρνει πολλές επιλογές, οι οποίες φαίνονται παρακάτω:

- `#` θα εφαρμόσει το Patch#
- `-p #` καθορίζει τον αριθμό των διαδοχικών υποκαταλόγων που θα αγνοήσει η εντολή patch(1).
- `-P` Η προκαθορισμένη ενέργεια είναι η εφαρμογή του `Patch` (ή `Patch0`). Αυτή η επιλογή κληρονομεί την προκαθορισμένη ενέργεια και θα απαιτήσει ένα `0` για να κάνει untar το κύριο αρχείο πηγαίου κώδικα. Αυτή η επιλογή είναι χρήσιμη σε ένα δεύτερο (ή μετέπειτα) `%patch` macro το οποίο απαιτούσε ένα διαφορετικό αριθμό από το πρώτο macro.
- Μπορείτε επίσης να γράψετε `%patch#` αντί για την πραγματική εντολή : `%patch # -P`

Αυτά πρέπει να είναι όλα τα macros που χρειάζεστε. Αφού τα έχετε αυτά σωστά, μπορείτε να κάνετε οποιαδήποτε εγκατάσταση θέλετε με `sh` σκριπτάκια. Οτιδήποτε συμπεριλάβετε μέχρι το `%build` macro (το οποίο θα συζητηθεί στο επόμενο τμήμα), εκτελείται μέσω `sh`. Κοιτάξτε το παραπάνω παράδειγμα για τους τύπους των πραγμάτων που μπορεί να θέλετε να κάνετε.

6.5 Χτίσιμο (Build)

Δεν υπάρχουν αλήθεια macros σε αυτό το τμήμα. Θα πρέπει απλά να βάλετε τις εντολές που θα χρειαστεί να χρησιμοποιήσετε για να χτίσετε το λογισμικό αφού έχετε ξεπακετάρει τον κώδικα, τον έχετε κάνει patch και έχετε μπει στον κατάλογο. Αυτό είναι απλά ένα ακόμη σύνολο εντολών που περνιούνται στο **sh**, έτσι ώστε όλες οι νόμιμες εντολές **sh** μπορούν να μπουν εδώ (συμπεριλαμβανομένων των σχολίων).

Ο τρέχων κατάλογος που δουλεύετε γίνεται reset σε κάθε ένα από αυτά τα τμήματα στο πιο ψηλό επίπεδο του καταλόγου που περιέχει τον κώδικα , να το έχετε υπόψη σας. Μπορείτε να μπείτε στους υποκαταλόγους με `cd` αν χρειαστεί.

6.6 Εγκατάσταση

Ούτε και εδώ υπάρχουν, πραγματικά, macros. Θέλετε βασικά να βάλετε ότι εντολές χρειάζονται για την εγκατάσταση. Αν έχετε στη διάθεσή σας το `make install` στο πακέτο που χτίζετε, βάλτε το εκεί. Αν όχι, μπορείτε είτε να κάνετε patch το makefile ώστε να γράψετε `make install` και στη συνέχεια να ζητήσετε να γίνει εδώ το `make install` , ή μπορείτε να τα εγκαταστήσετε εδώ με το χέρι χρησιμοποιώντας `sh` εντολές. Μπορείτε να θεωρήσετε τον τρέχων κατάλογο σαν το πιο ψηλό επίπεδο του καταλόγου με τον κώδικα.

6.7 Προαιρετικά σκριπτάκια πριν και μετά την Εγκατάσταση/Απεγκατάσταση

Μπορείτε να προσθέσετε σκριπτάκια που να τρέχουν πριν και μετά την εγκατάσταση/ απεγκατάσταση των εκτελέσιμων πακέτων. Ο κύριος λόγος γι'αυτό είναι να κάνετε πράγματα όπως να τρέξετε το `ldconfig` αφού εγκαταστήσετε ή αφαιρέσετε πακέτα που περιέχουν διαμοιραζόμενες βιβλιοθήκες. Τα macros για κάθε ένα από τα σκριπτάκια είναι τα παρακάτω:

- `%pre` είναι το macro που εκτελεί σκριπτάκια πριν την εγκατάσταση.
- `%post` είναι το macro που εκτελεί σκριπτάκια μετά την εγκατάσταση.
- `%preun` είναι το macro που εκτελεί σκριπτάκια πριν την απεγκατάσταση.
- `%postun` είναι το macro που εκτελεί σκριπτάκια μετά την απεγκατάσταση.

Τα περιεχόμενα αυτών των τμημάτων μπορεί να είναι οποιουδήποτε τύπου `sh` script, αν και *δε* χρειάζεστε το `#!/bin/sh`.

6.8 Αρχεία

Αυτό είναι το τμήμα όπου *πρέπει* να δώσετε τη λίστα των αρχείων για το εκτελέσιμο πακέτο. Το RPM δεν έχει τρόπο να γνωρίζει τι εκτελέσιμα εγκαταστάθηκαν σαν αποτέλεσμα του `make install`. *ΔΕΝ ΥΠΑΡΧΕΙ* τρόπος να το κάνετε αυτό. Μερικοί προτείνουν να εκτελέσετε τη `find` πριν και μετά την εγκατάσταση του πακέτου. Σε ένα πολυχρηστικό σύστημα, αυτό είναι απαράδεκτο καθώς είναι δυνατό να δημιουργηθούν άλλα αρχεία κατά τη διαδικασία χτισίματος ενός πακέτου, τα οποία δεν έχουν απολύτως καμία σχέση με το ίδιο το πακέτο.

Υπάρχουν μερικά διαθέσιμα macros που κάνουν τέτοιες ειδικές λειτουργίες. Αυτά είναι τα παρακάτω :

- `%doc` χρησιμοποιείται για να μαρκάρει μέσα στο πακέτο πηγαίου κώδικα την τεκμηρίωση που θέλετε να εγκατασταθεί με μία εγκατάσταση binary. Τα κείμενα θα εγκατασταθούν στο `/usr/doc/$NAME-$VERSION-$RELEASE`. Μπορείτε να έχετε με αυτό το macro λίστα πολλαπλών κειμένων στη γραμμή εντολών, ή να πάρετε τη λίστα καθενός ξεχωριστά χρησιμοποιώντας ένα macro για καθένα απ'αυτά.

- `%config` χρησιμοποιείται για να μαρκάρει τα αρχεία ρυθμίσεων σε ένα πακέτο. Αυτό περιλαμβάνει αρχεία όπως τα sendmail.cf, passwd, κλπ. Αν αργότερα απεγκαταστήσετε ένα πακέτο που περιέχει αρχεία ρυθμίσεων, όποια αρχεία δεν άλλαξαν θα σβηστούν και όποια έχουν υποστεί αλλαγές θα μετονομαστούν στο παλιό όνομα με κατάληξη `.rpmsave` στο όνομα του αρχείου. Μπορείτε, επίσης, να πάρετε τη λίστα πολλαπλών αρχείων με αυτό το macro.
- `%dir` μαρκάρει ένα μόνο κατάλογο σε μια λίστα αρχείων που θα συμπεριληφθεί σαν να ανήκει στο πακέτο. Είναι προκαθορισμένο, ότι αν δείτε τη λίστα ενός καταλόγου *ΧΩΡΙΣ* κάποιο `%dir` macro, *ΟΛΑ* σε αυτόν τον κατάλογο συμπεριλαμβάνονται στη λίστα αρχείων και εγκαθιστώνται σαν τμήμα του πακέτου.
- `%files -f` <`filename`> θα σας επιτρέψει να εμφανίσετε λίστα των αρχείων σας σε κάποιο αυθαίρετο αρχείο μέσα στον κατάλογο του πηγαίου κώδικα όπου γίνεται το χτίσιμο. Αυτό είναι καλό σε περιπτώσεις όπου έχετε ένα πακέτο που μπορεί να χτίσει τη δική του λίστα αρχείων. Τότε απλά ενσωματώνετε εδώ αυτή τη λίστα αρχείων και δε χρειάζεται να εμφανίσετε ειδική λίστα αρχείων.

Η μεγαλύτερη παγίδα στη λίστα αρχείων είναι η εμφάνιση καταλόγων. Αν εμφανίσετε το `/usr/bin` κατά λάθος, το εκτελέσιμο πακέτο σας θα περιέχει *κάθε* αρχείο στο `/usr/bin` στο σύστημά σας.

6.9 Χτίζοντας το

6.9.1 Ο δενδροειδής κατάλογος του πηγαίου κώδικα

Το πρώτο που χρειάζεστε είναι ένα καλά ρυθμισμένο δένδρο για χτίσιμο (build tree). Αυτό ρυθμίζεται χρησιμοποιώντας το αρχείο `/etc/rpmrc`. Οι περισσότεροι θα χρησιμοποιήσουν απλά το `/usr/src`.

Μπορεί να χρειαστεί να δημιουργήσετε τους παρακάτω καταλόγους για να φτιάξετε ένα δένδρο για χτίσιμο (build tree):

- `BUILD` είναι ο κατάλογος όπου γίνεται το χτίσιμο από το RPM. Δε χρειάζεται να τεστάρετε το χτίσιμο κάπου συγκεκριμένα, αλλά εδώ θα κάνει το RPM θα κάνει το χτίσιμό του.
- `SOURCES` είναι ο κατάλογος όπου θα πρέπει να βάλετε τα αρχικά tar αρχεία πηγαίου κώδικα και τα patches. Εδώ είναι που έχει προκαθοριστεί να ψάξει το RPM.
- `SPECS` είναι ο κατάλογος όπου πρέπει να πάνε όλα τα αρχεία προδιαγραφών (spec).
- `RPMS` είναι εκεί όπου θα βάλει το RPM όλα τα εκτελέσιμα RPMs μετά το χτίσιμο.
- `SRPMS` είναι εκεί που θα μπουν όλα τα RPMs πηγαίου κώδικα.

6.9.2 Έλεγχος στο χτίσιμο

Το πρώτο πράγμα που πιθανά να θέλετε να κάνετε είναι να πάρετε τον κώδικα και να χτίσετε το πακέτο χωρίς να χρησιμοποιήσετε καθόλου το RPM. Για να το κάνετε αυτό, ξεπακετάρετε τον πηγαίο κώδικα, και αλλάξτε το όνομα του καταλόγου σε $NAME.orig. Ξεπακετάρετε τον κώδικα ξανά. Χρησιμοποιήστε τον κώδικα για να χτίσετε το πακέτο. Πηγαίνετε στον κατάλογο με τον κώδικα και ακολουθήστε τις οδηγίες για το χτίσιμο. Αν πρέπει να αλλάξετε πράγματα, θα χρειαστείτε ένα μπάλωμα (patch). Μόλις το χτίσετε, καθαρίστε τον κατάλογο με τον πηγαίο κώδικα. Σιγουρευτείτε ότι διαγράψατε όλα τα αρχεία που δημιουργήθηκαν από το script `configure`. Στη συνέχεια, βγείτε από τον κατάλογο με τον πηγαίο κώδικα στον πιο πάνω γράφοντας `cd ..` Κάντε κάτι σαν το παρακάτω:

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

Αυτό θα σας δημιουργήσει ένα patch το οποίο μπορείτε να χρησιμοποιήσετε στο αρχείο προδιαγραφών σας (spec file). Σημειώστε ότι το "linux" που βλέπετε στο όνομα του patch είναι απλά αναγνωριστικό. Μπορείτε να χρησιμοποιήσετε κάτι πιο επεξηγηματικό όπως "config" ή "bugs" ώστε να εξηγήσετε *γιατί* χρειάστηκε να φτιάξετε αυτό το patch. Είναι επίσης καλή ιδέα να κοιτάξετε το patch αρχείο που φτιάχνετε πριν να το χρησιμοποιήσετε ώστε να σιγουρευτείτε ότι δεν συμπεριλαμβάνονται κατά λάθος τυχόν εκτελέσιμα.

6.9.3 Δημιουργώντας τη λίστα αρχείων

Τώρα που έχετε τον κώδικα για το χτίσιμο του πακέτου και ξέρετε πώς θα το κάνετε, χτίστε το και εγκαταστήστε το. Κοιτάξτε την έξοδο της διαδικασίας εγκατάστασης και χτίστε τη λίστα αρχείων σας από αυτή που θα χρησιμοποιήσετε στο αρχείο προδιαγραφών. Συνήθως, χτίζουμε το αρχείο προδιαγραφών παράλληλα με όλα αυτά τα βήματα. Μπορείτε να δημιουργήσετε το αρχικό και να συμπληρώσετε τα εύκολα τμήματα, και στη συνέχεια να συμπληρώνετε και τα υπόλοιπα βήματα καθώς προχωράτε.

6.9.4 Χτίζοντας το Πακέτο με RPM

Από τη στιγμή που έχετε ένα αρχείο προδιαγραφών (spec file), είστε έτοιμοι να δοκιμάσετε να χτίσετε το πακέτο σας. Ο πιο χρήσιμος τρόπος να το κάνετε είναι με μια εντολή σαν την ακόλουθη :

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

Υπάρχουν επίσης και άλλες χρήσιμες επιλογές με το switch `-b` :

- `p` σημαίνει να τρέξει απλά το τμήμα `prep` του specfile.
- `l` είναι ένας έλεγχος λίστας που κάνει μερικούς ελέγχους στα `%αρχεία`.
- `c` κάνει ότι λέει το τμήμα prep και μεταγλωττίζει. Αυτό είναι χρήσιμο όταν δεν είστε σίγουροι για το αν ο κώδικάς σας θα γίνει build. Φαίνεται άχρηστο επειδή μπορεί να θέλετε να συνεχίσετε να παίζετε με τον ίδιο τον κώδικα μέχρι να χτιστεί και μετά να χρησιμοποιήσετε το RPM, αλλά όταν εξοικειωθείτε με τη χρήση του RPM, θα βρείτε στιγμές που θα θελήσετε να το χρησιμοποιήσετε.
- `i` κάνει ότι λέει το τμήμα prep, μεταγλωττίζει και εγκαθιστά.
- `b` κάνει ότι λέει το τμήμα prep, μεταγλωττίζει, εγκαθιστά και χτίζει μόνο το εκτελέσιμο πακέτο.
- `a` τα χτίζει όλα (και τα εκτελέσιμα και τα πηγαίου κώδικα πακέτα).

Υπάρχουν αρκετές παραλλαγές στο `-b` switch. Αυτές είναι οι ακόλουθες :

- `--short-circuit` θα μεταβεί κατευθείαν σε ένα συγκεκριμένο σημείο (μπορεί να χρησιμοποιηθεί μόνο με τα c και i).
- `--clean` απομακρύνει το δένδρο χτισίματος (build tree) όταν τελειώσει.
- `--keep-temps` θα κρατήσει όλα τα temp αρχεία και τα scripts που φτιάχτηκαν στο /tmp. Μπορείτε να δείτε ποια αρχεία δημιουργήθηκαν στον κατάλογο /tmp χρησιμοποιώντας την επιλογή `-v` .
- `--test` δεν εκτελεί πραγματικά τα βήματα, μόνο ελέγχει αν μπορούν να γίνουν.

6.10 Ελέγξτε το

Αφού φτιάξατε ένα εκτελέσιμο και ένα με τον πηγαίο κώδικα rpm για το πακέτο σας, χρειάζεται να το ελέγξετε. Ο ευκολότερος και καλύτερος τρόπος είναι να χρησιμοποιήσετε ένα τελείως διαφορετικό μηχάνημα για να το δοκιμάσετε. Στο κάτω-κάτω, έχετε κάνει τόσα `make install` στο μηχάνημά σας, που θα πρέπει να εγκατασταθεί καλά.

Μπορείτε να ελέγξετε το πακέτο γράφοντας `rpm -u packagename`, αλλά αυτό μπορεί να σας εξαπατήσει επειδή στο χτίσιμο του πακέτου, κάνατε `make install`. Αν αφήσετε κάτι έξω από τη λίστα αρχείων σας, δε θα απεγκατασταθούν. Θα εγκαταστήσετε ξανά το εκτελέσιμο πακέτο (binary package) και το σύστημά σας θα λειτουργεί και πάλι, αλλά το rpm δεν είναι εντάξει. Σιγουρευτείτε και έχετε υπόψη σας ότι μπορεί εσείς να γράφετε `rpm -ba package`, πολλοί όμως εγκαθιστούν το πακέτο σας κάνοντας `rpm -i package`. Σιγουρευτείτε ότι δεν κάνετε τίποτα στα τμήματα `build` ή `install` που θα χρειαστεί να γίνει όταν τα εκτελέσιμα πακέτα εγκαθίστανται μόνα τους.

6.11 Τι να κάνετε με τα νέα σας RPMs

Μόλις φτιάξετε το δικό σας νέο RPM για κάτι (υποθέτουμε ότι είναι κάτι που δεν υπάρχει σε RPM), μπορείτε να προσφέρετε τη δουλειά σας και σε άλλους (επίσης υποθέτουμε ότι φτιάξατε RPM ενός προγράμματος που διανείμεται ελεύθερα). Για να το κάνετε αυτό, θα θελήσετε να το ανεβάσετε στο *ftp.redhat.com* `<ftp://ftp.redhat.com>`.

6.12 Και τώρα;

Διαβάστε παρακαλώ τα παραπάνω τμήματα σχετικά με τον Έλεγχο και του τι να κάνετε με τα νέα RPMs. Θέλουμε όσα RPMs είναι δυνατό να έχουμε, και θέλουμε μάλιστα να είναι και καλά RPMs. Παρακαλώ δώστε αρκετό χρόνο στο τεστάρισμα των πακέτων RPMs και κάντε τον κόπο να τα βάλετε στο internet για το καλό όλων μας. Επίσης, *παρακαλώ* σιγουρευτείτε ότι κάνετε upload *ελεύθερα διαθέσιμο λογισμικό*. Εμπορικό και shareware λογισμικό *δε* θα πρέπει να γίνεται upload εκτός κι αν έχουν αναφέρεται ρητά ότι επιτρέπεται στην άδεια πνευματικής ιδιοκτησίας. Αυτό περιλαμβάνει το λογισμικό Netscape, ssh, pgp, κλπ.

7 Χτίζοντας RPM για πολλές αρχιτεκτονικές

Το RPM μπορεί να χρησιμοποιηθεί για να χτίσετε πακέτα για Intel i386, Digital Alpha που τρέχουν Linux και σε Sparc. Έχει αναφερθεί ότι δουλεύει επίσης και σε σταθμούς εργασίας SGI και HP. Υπάρχουν διάφορα χαρακτηριστικά που κάνουν το χτίσιμο των πακέτων εύκολο σε όλες τις πλατφόρμες. Το πρώτο είναι η ντιρεκτίβα "optflags" στο `/etc/rpmrc`. Μπορεί να χρησιμοποιηθεί για να θέσετε flags που χρησιμοποιούνται όταν χτίζετε λογισμικό, σε τιμές που εξαρτώνται από την αρχιτεκτονική του συστήματος. Άλλο ένα χαρακτηριστικό είναι τα "arch" macros στο αρχείο προδιαγραφών (spec file). Μπορούν να χρησιμοποιηθούν για να κάνετε διάφορα πράγματα ανάλογα με την αρχιτεκτονική του συστήματος στην οποία το χτίζετε. Άλλο ένα χαρακτηριστικό είναι η ντιρεκτίβα "Exclude" στην επικεφαλίδα (header).

7.1 Ένα απλό αρχείο προδιαγραφών (spec File)

Το παρακάτω είναι τμήμα του αρχείου προδιαγραφών (spec file) για το πακέτο "fileutils". Είναι ρυθμισμένο για να χτίζεται και σε Alpha και σε 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

Σε αυτό το παράδειγμα, μπορείτε να δείτε πώς χρησιμοποιείται η ντιρεκτίβα "optflags" από το `/etc/rpmrc`. Ανάλογα με το πάνω σε ποια αρχιτεκτονική θα χτίσετε, δίνετε την κατάλληλη τιμή στο `RPM_OPT_FLAGS`. Πρέπει να κάνετε patch το Makefile έτσι ώστε το πακέτο σας να χρησιμοποιεί αυτή τη μεταβλητή αντί των κανονικών ντιρεκτίβων που θα μπορούσατε να χρησιμοποιήσετε (όπως `-m486` και `-O2`). Μπορείτε να πάρετε μια καλή ιδέα για το τι χρειάζεται να κάνετε εγκαθιστώντας το πηγαίο κώδικα του πακέτου και αποσυμπιέστε τον κώδικα και εξετάστε το Makefile. Κοιτάξτε τότε το patch για το Makefile και δείτε τι αλλαγές πρέπει να κάνετε.

7.3 Macros

Το `%ifarch` macro είναι πολύ σημαντικό σε όλα αυτά. Τις περισσότερες φορές θα χρειαστεί να εφαρμόσετε ένα-δύο patch ειδικά για μια συγκεκριμένη αρχιτεκτονική. Σε αυτή την περίπτωση, το RPM θα σας επιτρέψει να εφαρμόσετε αυτό το patch σε μία αρχιτεκτονική μόνο.

Στο παραπάνω παράδειγμα, το fileutils έχει ένα patch για μηχανές 64 bit. Προφανώς, αυτό θα πρέπει να εφαρμοστεί μόνο σε Alpha για την ώρα. Έτσι, προσθέτουμε ένα `%ifarch` macro γύρω από το 64 bit patch όπως παρακάτω:

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

Αυτό εξασφαλίζει ότι το patch δε θα εφαρμοστεί σε οποιαδήποτε αρχιτεκτονική εκτός από alpha.

7.4 Αφαιρώντας αρχιτεκτονικές από Πακέτα

Για να μπορείτε να συντηρείτε πακέτα RPMs πηγαίου κώδικα σε έναν κατάλογο για όλες τις πλατφόρμες, έχουμε υλοποιήσει την ικανότητα να "αφαιρείτε" πακέτα ώστε να μη χτιστούν σε συγκεκριμένες αρχιτεκτονικές. Αυτό γίνεται ώστε να μπορείτε να κάνετε ακόμη πράγματα όπως

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

και να χτίσετε τα σωστά πακέτα. Αν δεν έχετε μεταφέρει μια εφαρμογή σε μια συγκεκριμένη πλατφόρμα, το μόνο που πρέπει να κάνετε είναι να προσθέσετε μια γραμμή σαν αυτή :

```
ExcludeArch: axp
```

στην επικεφαλίδα του αρχείου προδιαγραφών του πακέτου με τον κώδικα. Στη συνέχεια, ξαναχτίστε το πακέτο στην πλατφόρμα για την οποία προορίζεται. Θα έχετε τότε ένα πακέτο πηγαίου κώδικα που παίζει σε Intel και μπορεί εύκολα να μεταφερθεί σε Alpha.

7.5 Τελειώνοντας

Συνήθως, είναι ευκολότερο να χρησιμοποιήσετε RPM για να φτιάξετε πακέτα για πολλαπλές -αρχιτεκτονικές από το να πάρετε το κάθε πακέτο ξεχωριστά και να το χτίσετε στα δύο μέρη. Καθώς πολλά από τα μεγάλα πακέτα χτίζονται, αυτό γίνεται ευκολότερο, παρόλαυτά. Όπως πάντα, ο καλύτερος τρόπος για να αναζητήσετε βοήθεια όταν κολλήσετε χτίζοντας ένα RPM είναι να δείτε τον κώδικα ενός παρόμοιου πακέτου.

8 Πνευματικά Δικαιώματα

9 Περί της Ελληνικής Μετάφρασης

Για σχόλια, παρατηρήσεις, διορθώσεις, ενημερώσεις περί της Ελληνικής μετάφρασης αυτού του HOWTO, στείλτε email στην Βούλα Σανιδά : `voulariba@hellug.gr`