Handleiding voor het maken van rpm bestanden voor Mandrakelinux.

Door: Marco Meijer e-mail: korrel@NOSPAM.mandrakeclub.nl
9 juni 2004, versie 0.23

Wat je eerst moet weten.

Wat is een RPM bestand: Een simpele RPM is niets anders dan een pakketje bestanden en een beschrijving van waar ze moeten komen.
Daardat de plaats van de bestanden per distributie iets kunnen verschillen, en de afhankelijkheden in verschillende distributies over verschillend genoemde pakketten zijn verdeeld, werkt een RPM dat voor bijvoorbeeld suse geschreven is vaak slecht in bijvoorbeeld Mandrake omdat het de bestanden dan op de verkeerde plaatsen zet.

Deze handleiding is bedoeld voor iedereen die zelf rpm bestanden wil maken.
Het grootste deel van het maken van een rpm bestand bestaat uit het schrijven van een SPEC bestand. Dit SPEC bestand is een script dat er voor zorgt dat materiaal wat je wilt installeren op de juiste plaats en in het juiste formaat wordt geplaats in je distributie. Dit materiaal kan bestaan uit broncode maar ook ook plaatjes, geluidsfragmenten, binaries etc... In het SPEC geef je alles aan dat van belang is voor het maken van een goed RPM bestand.

Nou zal het schrijven van een script voor sommigen overkomen als moeilijk, maar dat valt in dit geval best mee.
Het grootste deel van je SPEC bestand kun je gewoon kopieren en plakken. Verder is het een mooie kans om iets van de basis van het schrijven van scripts te weten te komen, iets wat in linux altijd van pas komt.

Wat het je nodig voor het maken van een rpm bestand.

Enige kennis van de mappenstructuur van linux. (basiskennis)

En de volgende bestanden op je computer :

rpm	(staat al op je computer)
rpmbuild	(rpm-build-4.2.2-7mdk)
spec-helper	(spec-helper-0.10.1-1mdk)
libtool	(libtool-1.4.3-10mdk)

Een nieuwere versie van deze bestanden is natuurlijk ook goed.

Als de programma's geïnstalleerd zijn, moeten we nog wat voorbereidend werk doen.
We moeten een werkmap maken in je homemap voor de rpms.
Start hiervoor de konsole op en geef het volgende commando in je home directory:

mkdir -p ~/rpm/{BUILD,RPMS/i586,RPMS/noarch,SOURCES,SRPMS,SPECS,tmp}

Je krijgt nu een rpm map met de volgende structuur.

~/rpm/BUILD Hier worden de source files uitgepakt en omgezet naar binaries

~/rpm/RPMS Hier komt uiteindelijk je RPM uit.(hier gaat het uiteindelijk om)

~/rpm/RPMS/i586 De map voor de rpm bestanden voor i586

~/rpm/RPMS/noarch De map voor de rpm bestanden zonder bijzondere architectuur vereisten.

~/rpm/SOURCES De source (*.tar.bz2).

~/rpm/SPECS De plaats van je SPEC scripts.

~/rpm/SRPMS Dit SRPM bestand bevat na het maken van de rpm de oorspronkelijke broncode bestanden en het SPEC bestand.

~/rmp/tmp Hier bouwt linux je tijdelijke structuur op voordat hij er een RPM bestand van maakt.

controleer of al deze mappen er zijn.

Nu moet je nog één ding doen voordat je kunt beginnen en dat is twee configuratie bestanden maken in je
homemap.

.rpmrc

buildarchtranslate: i386: i586
buildarchtranslate: i486: i586
buildarchtranslate: i586: i586
buildarchtranslate: i686: i586

.rpmmacros

%_topdir /home/naam/rpm
%_tmppath /home/naam/rpm/tmp
%distribution Mandrakelinux
%vendor Mandrakesoft

Vervang /home/naam/ voor de plaats van je eigen homemap .

Tijd voor het echte werk:

Je hebt nu alle voorbereidingen gedaan het is nu tijd voor het maken van je eerste rpm.

Je begint eerst even heel simpel. Stel je wilt een achtergrond installeren in de map
/usr/share/wallpapers
zodat hij beschikbaar wordt voor algemeen gebruik.
Dit is mogelijk met de volgende SPEC file.
(het plaatje mooiplaatje.jpg moet zich bevinden in de SOURCES map.)

Name: mooiplaatje
Summary: wallpaper voor algemeen gebruik
Version: 1
Release: 1mcnl
Source0: mooiplaatje.jpg
Group: Multimedia
License: GPL
BuildRoot: %{_tmppath}/mooiplaatje-1-buildroot

%description
Een kleine omschrijving hoe het plaatje eruitziet

%install
rm -rf %{buildroot}
install -m644 %{SOURCE0} -D %{buildroot}/usr/share/wallpapers/mooiplaatje.jpg

%clean
rm -rf %{buildroot}

%files
/usr/share/wallpapers/mooiplaatje.jpg

%changelog

** Fri May 28 2004 Marco Meijer* <korrel@mandrakeclub.nl> *1-1mdk*
- eerste versie

Dit SPEC bestand doet zo ongeveer het meest simpele mogelijk en doet niet anders dan het bestand mooiplaatje.jpg verplaatsen naar de map */usr/share/wallpapers/.*

Maar wat heb ik nu allemaal ingevuld. (van boven naar beneden)

Name:	spreekt voor zich, de naam van het bestand.
Summary:	een zeer korte beschrijving van de inhoud.
Version:	het versienummer van het bronbestand
Release:	Is het interne versienummer. Dus wanneer ik een wijzigingen maak aan mijn rpm maar met het zelfde source bestand. (Als de eerste versie van je rpm niet goed werkt bijvoorbeeld) dan kun je dit verhogen met 1 zodat het bij installatie gezien wordt als een nieuwere versie. De mcNL staat voor een mandrakeclub NL rpm zodat je kunt zien waar je hem vandaan heb.
Source0:	Dit is de naam van je bron bestanden in dit geval *mooiplaatje.jpg*
Group:	Dit is van belang voor de indeling van bijvoorbeeld rpmdrake hierover later meer.
License:	De licentie waaronder een bestand valt. (ZORG DAT DIT KLOPT, BELANGRIJK!)
BuildRoot:	%{_tmppath}/mooiplaatje-buildroot Dit zorgt ervoor dat de bron niet echt wordt geïnstalleerd maar binnen onze tijdelijk mappen blijft. Deze zin is dan ook erg belangrijk. Gewoon toevoegen.
%description:	Hierna volgt een uitgebreide omschrijving van de RPM. (hou het duidelijk)

%install
rm -rf %{buildroot}
install -m644 %{SOURCE0} -D %{buildroot}/usr/share/wallpapers/mooiplaatje.jpg

Ok, dit is de regel waar het in deze SPEC om draait. Wat hier staat is nu eigenlijk:

De eerste regel is altijd hetzelfde en zorgt ervoor dat eventueel aanwezige oudere versies verwijderd worden.
In de tweede regel staat: installeer het bestand uit {SOURCE0} in de map /usr/share/wallpapers/, noem het mooiplaatje.jpg, maak hiervoor alle benodige eventueel ontbrekende mappen aan en stel de juiste rechten in.

(-m644 geeft de rechten op. In dit geval worden de rechten voor de eigenaar op kezen en schrijven gezet, en die voor de groep en voor iedereen op lezen. Als je hier meer over wil weten kan je in de console "man chmod" invoeren, dan krijg je een omschrijving)

de tekst *%{buildroot}* staat voor je root directory. Omdat we deze met het configuratie bestandje naar de tpm map hadden omgeleid zal hij de binaries tijdens het bouwen van de rpm daar installeren. Terwijl de

RPM straks zal installeren in echte root map.

%clean
rm -rf %{buildroot}

Deze regel is altijd hetzelde en zorgt ervoor dat alles weer netjes schoon wordt achtergelaten als de RPM klaar is voor je volgende project. (Gewoon toevoegen dus)

%files
/usr/share/wallpapers/mooiplaatje.jpg

Ok dit is een lijst van alle bestanden die uiteindelijk worden toegevoegd bij het installeren van de rpm (en bij het verwijderen weer verwijderd worden). Het is van belang dat deze lijst klopt anders zal die het build proces afbreken. Als dit gebeurt, zal er wel precies verteld worden wat er niet klopt zodat je dit kunt aanpassen, goed lezen dus!
(wat ik meestal doe als ik niet precies weet wat er allemaal voor bestanden in zitten, is gewoon niks invullen, gewoon het script uitvoeren en aan het eind lezen wat ik allemaal moet toevoegen.)

%changelog

** Fri May 28 2004 Marco Meijer* <korrel@mandrakeclub.nl> *1-1mdk*
- eerste versie

Ok, eind van dit voorbeeld SPEC bestand, de changelog. Deze bevat alle wijzigingen sinds de vorige versie. Je bent niet verplicht iets in te vullen, maar het is wel erg handig om er gebruik van te maken. Gewoon bovenstaande notering aanhouden.

Je hebt een script in de map SPECS en een mooiplaatje.jpg in de map SOURCES, wat nu?

Dit is eigenlijk heel simpel:
Je gaat naar de konsole, de map /rpm/SPECS
en daar geef je het volgende commando:

rpm -ba mooiplaatje.spec --clean

Als alles goed werkt worden er nu twee bestanden aangemaakt en is je RPM klaar.

%{buildroot}/rpm/RPMS/i586/mooiplaatje-1-1mdk.i586.rpm
%{buildroot}/rpm/SRPMS/mooiplaatje-1-1mdk.src.rpm

Als dit niet zo is kijk dan of je een error krijgt en wat er fout is (vaak leer je het meest van de foutmeldingen.)

Een uitleg over het gebruiken van het %define commando.

Dit is natuurlijk een heel simpel SPEC bestand, maar je zal later zien dat je niet zo veel hoeft te veranderen om een iets ingewikkelder RPM te maken.
Wat je nu gaat doen is het commando %define gebruiken, dit heb ik tot nu toe vermeden om het niet ingewikkelder te laten lijken dan het is.
Het commando %define houdt in dat je een bepaalde waarde toekent aan een naam. Dit maakt het makkelijker de SPEC te hergebruiken, omdat je niet steeds overal het woord "mooiplaatje" hoeft te vervangen.

Ook zijn er een aantal %define instellingen die je standaard kunt gebruiken en je dus niet hoeft op te geven.
Je hoeft deze natuurlijk niet uit je hoofd te weten maar hou dit lijstje bij de hand.
In kleur zijn de meest voorkomende aangegeven.

%{_tmppath}	/home/???/rpm/tmp #de plaats van je tijdelijke tmp map.
%{_vendor}	MandrakeSoft
%{_os}	linux
%{_prefix}	/usr
%{_exec_prefix}	%{_prefix}
%{_bindir}	%{_exec_prefix}/bin
%{_sbindir}	%{_exec_prefix}/sbin
%{_libexecdir}	%{_exec_prefix}/libexec
%{_datadir}	%{_prefix}/share
%{_gamesdir}	games
%{_gamesbindir}	%{_prefix}/%{_gamesdir}
%{_gamesdatadir}	%{_datadir}/%{_gamesdir}
%{_menudir}	%{_prefix}/lib/menu
%{_iconsdir}	%{_datadir}/icons
%{_miconsdir}	%{_datadir}/icons/mini
%{_liconsdir}	%{_datadir}/icons/large
%{_sysconfdir}	/etc
%{_sharedstatedir}	%{_prefix}/com
%{_localstatedir}	/var
%{_lib}	lib
%{_libdir}	%{_exec_prefix}/%{_lib}
%{_includedir}	%{_prefix}/include
%{_oldincludedir}	/usr/include
%{_infodir}	/usr/share/info
%{_mandir}	/usr/share/man
%{_initrddir}	%{_sysconfdir}/rc.d/init.d
%{_defaultdocdir}	%{_usr}/share/doc
%{_libexecdir}	%{_libdir}
%{_localstatedir}	%{_var}/lib
%{_sysconfdir}	/etc

Ons vorige voorbeeld had er met gebruik van %define zo uit gezien.

```
%define name mooiplaatje
%define summary wallpaper voor algemeen gebruik
%define version 1
%define release 1mcnl

Name:           %{name}
Summary:        %{summary}
Version:        %{version}
Release:        %{release}
Source0:        %{name}.jpg
Group:          Multimedia
License:        GPL
BuildRoot: %{_tmppath}/%{name}-%{version}-buildroot

%description
Een kleine omschrijving hoe het plaatje eruitziet

%install
rm -rf %{buildroot}
install -m644 %{SOURCE0} -D %{buildroot}/%{_datadir}/wallpapers/%{name}.jpg
%clean
rm -rf %{buildroot}

%files
%{_datadir}/*

%changelog
```

** Fri May 28 2004 Marco Meijer <korrel@mandrakeclub.nl> 1-1mdk*
- eerste versie

Zoals je ziet kunnen we het script nu met wat kleine wijzigingen aanpassen voor een plaatje met een andere naam.
voor dit kleine script lijkt het natuurlijk wat omslachtig, maar wanneer het script langer wordt kan het een heleboel tijd gaan besparen.

Genoeg basis kennis, tijd voor het verpakken van een programma, daar gaat het tenslotte om.

Voor dit deel van de handleiding ga ik er van uit dat je al eens een programma gecompileerd hebt in linux (./configure, make, makeinstall). En dan weet je ook dat je hiervoor vaak enige dependencies moet installeren.
De devel pakketen zijn meestal die dependencies. Welke dit zijn is meestal te achterhalen via de website van het programma en anders via het lezen van de errors die je krijgt tijdens het compileren.

Voor dit voorbeeld moeten het volgende pakket zijn geinstalleerd.
libxfree86-devel, libqt3-devel en libkdebase4-devel

Het voorbeeld dat we nu gaan maken is het pakket knetworkled.
Een programma dat via een icoontje aangeeft of er netwerkverkeer is in de takenbalk.

De homepage waar we de source kunnen downloaden bevindt zich op:
http://www.glitch13.com/knetworkled.php

Hier kunnen we ook zien dat het programma gemaakt is door:
brent <glitch13@glitch13.com>

Er is een engelse omschrijving wat het programma is:
"KDE Network activity monitory, behaves just like the windows system tray (systemtray) icon that blinks on network activitiy."

De licentie is LGPL.

Deze gegevens kun je later allemaal invullen in je SPEC bestand.

Je uiteindelijke SPEC bestand komt er ongeveer zo uit te zien:

```
%define __libtoolize /bin/true
%define name    knetworkled
%define version 0.5
%define release 1mcnl
%define summary KDE Network activiity monitory
%define section System/Monitoring

Name:         %{name}
Summary:      %{summary}
Version:      %{version}
Release:      %{release}
Vendor:       brent  <glitch13@glitch13.com>
URL:          http://www.glitch13.com/knetworkled.php
License:      LGPL
Group:        Network
Packager:     Marco Meijer <korrel@mandrakeclub.nl>

Source0:      %{name}-%{version}.tar.bz2

BuildRoot:    %{_tmppath}/%{name}-%{version}-%{release}-buildroot
```

```
BuildRequires:  libxfree86-devel libqt3-devel libkdebase4-devel

%description
KDE Network activiity monitory, behaves just like the windows system tray (systemtray) icon that blinks on network activitiy.

%prep

%setup -q

%build

%configure
%make

%install
rm -rf %{buildroot}
%makeinstall

#menu
mkdir -p %{buildroot}/%{_menudir}
cat > %{buildroot}/%{_menudir}/%name << EOF
?package(%name): \
command="%{_bindir}/%name" \
needs="X11" \
icon="%name.png" \
section="%section" \
title="%name" \
longtitle="%summary"
EOF

%post
%update_menus

%postun
%clean_menus

%clean
rm -rf %{buildroot}

%files
%defattr(-,root,root,0755)
%doc README NEWS COPYING AUTHORS
%_{menudir}/*
%{_iconsdir}/*
%{_bindir}/%{name}
%{_datadir}/applnk/Utilities/knetworkled.desktop
%{_datadir}/knetworkled/knetworkledui.rc
%{_datadir}/doc/HTML/en/*

%changelog
*Sun May 30 2004 Marco Meijer <korrel@mandrakeclub.nl> mcnl
new release
```

Zoals je ziet is er veel hetzelfde als in het eerste voorbeeld.
Hieronder volgt een uitleg van de nog niet besproken onderdelen.

```
%define __libtoolize /bin/true
```

Deze eerste regel is om een "build error""te voorkomen. anders krijg je de volgende error bij hij maken van een rpm bestand dat van kde gebruik maakt.
unrecognized option `--tag=CXX' Try `libtool --help' for more information
Dit is bekend probleem bij Mandrake en komt door dat mandrake libtool-1.4 gebruikt en kde libtool-1.5 allemaal niet echt belangrijk dus gewoon toevoegen.

```
Vendor:         brent  <glitch13@glitch13.com>
```

Naam en email adres van de maker van het programma. Stond op de website dus gewoon invullen.

URL: *http://www.glitch13.com/knetworkled.php*

Homepage van het programma handig voor het vinden van informatie of voor het vinden van een nieuwere versie

Packager: *Marco Meijer <korrel@mandrakeclub.nl>*

Hier kun je je eigen naam invullen. De naam van de maker van de rpm zodat gebruikers van je pakket het je kunnen melden als er wat mis is of omdat ze iets anders kwijt willen.

BuildRequires: *libxfree86-devel libqt3-devel libkdebase4-devel*

Hier kun je de paketten invullen die nodig zijn om de rpm te bouwen je kunt deze regel ook weglaten maar is erg handig als iemand anders het pakket later nog een keer wil maken als er bijvoorbeeld nieuwe versie uit is.

In de volgende regels word je broncode omgezet naar een binary bestand.

De *%prep* regel maakt een tijdelijke map aan in je build directory
%setup -q Pakt de source uit in je nieuwe tijdelijke builld directory.
%build

De *%configure* regel doet een beetje hetzelfde wat ./configure doet maar weet alle voorkeurinstellingen van mandrake al dus die hoef je niet meer in te vullen.
Het *%make* commando je raad het waarschijnlijk al is niet anders dan een uitgebreid make commando.

Als je SPEC script dit punt voorbij is dan gaat het goed.
Nu hoeft ie alleen nog maar de gemaakt bestanden op de juiste plaats te installeren in je tmp folder zodat ie er een rpm bestand van kan maken deze opdrachten staan na *%install.*

rm -rf %{buildroot}

Verwijdert eventueel achtergebleven map van een vorig poging indien aanwezig.

%makeinstall

Deze installeert de gegevens net als wanneer je make install doet maar dan niet in de root maar in je tijdelijke tmp map. Alle gegevens die nu daar in gezet worden eindigen later in de rpm. Behalve wanneer je expliciet zou aangeven dat je dat niet zou willen.

Nu zijn de binaries klaar. Je zou er nu een rpm bestand van kunnen maken maar aangezien dit bestand een uitvoerbaar bestand voor op je desktop is is het fijn als daar ook een icoonje voor in je menu wordt gezet. Hiervoor moet er een bestandje geplaatst worden in de %_menudir (/usr/lib/menu/)

mkdir -p %{buildroot}/%{_menudir}

Met deze opdracht maak je een mapstructuur (/usr/lib/menu/) aan in je tijdelijke tmp map.

Nu moet je een tekst bestandje maken in linux doe je dat met het commando cat.

cat > %{buildroot}/%{_menudir}/%name << EOF

Betekent dus bestandje maken in de map %_menudir met de naam van je bestand.
de inhoud van het bestand staat tussen *<<EOF* en *EOF*

*?package(%name): * = naam pakket
(dus niet naam icoon dus bij meerdere icoontjes is de package naam het zelfde)
*command="%{_bindir}/%name" * = uit te voeren commando in jouw geval wordt dit /usr/bin/knetworkled
*needs="X11" * = heeft X server nodig

*icon="%name.png" * = naam van het te gebruiken icoon dit kan een eigen icoon zijn, zoals in ons geval, maar kan ook al een bestaan icoon zijn dat al op je pc aanwezig is.

*section="%section" * = de plaats in de menu structuur in jouw geval System/Monitoring

*title="%name" * = naam van het programma voor in het menu

longtitle="%summary" = korte omschrijving van je programma

EOF

De menu gegevens zijn klaar, nu moet je alleen nog aangeven dat je het menu update bij installeren en schoonmaakt bij het verwijderen van je programma anders zou het icoon niet zichtbaar worden.

```
%post
%update_menus

%postun
%clean_menus
```

Deze opdracht zorgt ervoor dat dat gebeurd.

Ok basisvoorbereidingen zijn klaar nu moet je alleen nog gaan vertellen wat er in de rpm moet worden opgenomen.
dit start je met.

```
%files
```

Voor dat je beginnen met het plaatsen van de bestanden in je rpm moet je eerst de eigenaar ervan instellen.
Als je niks doet blijf jij de eigenaar en krijgt degene die installeert een foutmelding omdat jij (als gebruiker)
op zijn computer waarschijnlijk niet bestaat.) Ok hieronder volgt de opdracht zal in de meeste gevallen voldoen alleen bij speciale programmas zoals servers is dit soms anders.

```
%defattr(-,root,root,0755)
#  -: alle instellingen voor de bestanden blijven onveranderd.
# root: de eigenaar van je bestanden wordt root,
# root: de group van je bestanden wordt root ("users" is hier ook een veelgebruikte optie),
# 0755: alle instellingen voor de folders in dit pakket worden 0755 ( rwxr-xr-x ).
```

Dan kopieren we enkele kleine tekstbestanden naar de documentatie map.
Je moet maar even in de .tar kijken of deze bestanden aanwezig zijn. (Of gewoon de foutmelding afwachten)

```
%doc README NEWS COPYING AUTHORS
```

Nu het laatste stukje. In dit geval was dat het volgende:

```
%_{menudir}/*
%{_iconsdir}/*
%{_bindir}/%{name}
%{_datadir}/applnk/Utilities/knetworkled.desktop
%{_datadir}/knetworkled/knetworkledui.rc
%{_datadir}/doc/HTML/en/*
```

Als je niet weet wat erin zit is het beste niks invullen. dan geeft je rpm aan het eind van het maken een foutmelding en daar staat precies in welke bestanden je allemaal gemist heb.

Dit is in principe alles wat er voor dit programma nodig was om het SPEC script te maken.
Als je alles hebt ingevuld zou je het met. *rpm -ba knedworkled.spec --clean* je rpm bestand moeten kunnen maken.

Dit is einde van deze handleiding. Ik heb waarschijnlijk best wat gemist. Dit zal ik later aanvullen, dus als er nog opmerkingen en/of aanvullingen zijn, stuur die dan op! Er zijn veel alternatieve mogelijkheden om een SPEC bestand te maken/ De beste manier om er achter te komen wat handig is, is vaak om andere

SPECs te lezen. Deze zitten in de SRPM bestanden en krijg je door
rpm -i bestandnaam.SRPM te doen. De SPECs komen dan in de SPEC map en de SOURCE in de SOURCE map.
Veel succes!

Extra bronnen in engels:

Mandrake RPM HOWTO
http://qa.mandrakesoft.com/twiki/bin/view/Main/RpmHowTo

Redhat RPM to the max HOWTO
http://www.rpm.org/max-rpm/