http://clx.asso.fr/spip/?4-Quelques-exemples

4- Quelques exemples

- Documentations - Technique - RPM : Faites-le vous même ! -

Date de mise en ligne : mardi 25 janvier 2005

exemples vécus

Le RPM du fainéant

Auteur d'un jeu de tarot initialement pour Windows conçu sous Delphi de Borland, j'ai procédé à son transfert vers Linux à l'aide de Kylix (EDI Delphi sous Linux conçu par Borland. Kylix n'est pas libre. Il existe néanmoins une version "open edition" gratuite).

Dans un premier temps je l'ai diffusé sous la forme d'un tar.gz contenant les binaires et d'un autre contenant les sources. Je me suis alors lancé dans sa eRPéMisation et j'ai éclusé les howtos et autres sites persos sur le thème. Remarquons tout de suite celui de Mandriva (désormais disponible en Français).

Pas grand chose à la portée d'un anglophobe au vocabulaire limité à quelques dizaines de mots.
J'avais néanmoins compris quelques trucs :
– la structure obligatoire des répertoires à mettre en place ;
– l'existence d'un script d'*eRPéMisation* ;
– la possibilité de "tricher" sur des scripts existants en allant sur le CVS de Mandriva.

1) Les dossiers

Dans votre dossier personnel, vous avez créé les dossiers suivants :

– RPM

- RPM/SOURCES où il faut mettre les fichiers sources compressés
- RPM/SPECS où il faut créer son fichier de spécifications
- RPM/RPMS qui contient un dossier par plateforme
    - RPM/RPMS/i586 où sera créé votre RPM pour Pentium

- RPM/SRPMS où sera créé le "SRC.RPM" qui permet de refaire le RPM selon la méthode décrite plus bas
- RPM/tmp où sera stockée votre application compilée pendant la production du RPM

2) le fichier SPEC

C'est un fichier texte qui définit comment va être créé votre RPM. Celui qui suit est une version "Fainéant" ! Il ne compile rien mais se contente de recopier les fichiers à la manière d'un *tar.gz* à l'emplacement identique. J'ai fait cela car je n'arrivais pas à maîtriser le compilateur en ligne Kylix (entretemps, je me suis soigné) :

```
 _ %define name   bztarot
_ %define version 1.01
_ %define release 9
_ Name:          %{name}
_ Summary:       Jeu de tarot a 4 sous environnement graphique
_ Version:       %{version}
_ Release:       %{release}
_ License:       GPL
_ Group:         Games/Cards
_ URL:           http://www.beuselinck.com
_ Provides:      %{name} = %{version}-%{release}
_ Source:                bztarot-1.01.tar.bz2
_ %description
_ Ce jeu de tarot vous permet de jouer seul contre l'ordinateur
 dans des parties a 4. Respecte la reglementation de la Federation
 Francaise de Tarot.
_ %install
_ %post
_ %{update_menus}
_ %postun
_ %{clean_menus}
_ %files
_ /usr/bin/bztarot-1.01
_ /usr/bin/bztarot
_ /usr/lib/libborqt*
_ /usr/share/icons/bztarot.png
_ "/usr/share/applnk-mdk/More applications/Games/Cards/bztarot.desktop"
_ %changelog
_ * Mon Oct 12 2004 Vincent Beuselinck <vincent@beuselinck.com> 1.01
_ - ajout d'un écran d'accueil
_ - correction d'un bug (sigserv 11)
_ - rajout d'une entrée de menu oubliée
_ - compatibilité noyau 2.6
_ - aide integrée
_ - modifications pour empaquetage RPM
```

ATTENTION : Ce fichier SPEC n'est pas techniquement un bon modèle : il ne réalise pas la compilation mais se contente de recopier les binaires (Horreur !). L'entrée de menu qu'il crée a des résultats curieux selon les versions de MandrakeLinux

Dans un premier temps, j'avais accentué mes descriptions. Avec les commandes **urpmi** et *(rpm*, j'obtenais le résultat escompté mais dans rpmdrake les rubriques avec accent ou cédille disparaissaient. J'ai déposé un rapport de bogue et la réponse a été en substance qu'il fallait faire ça en UTF8 (???) pour que ça marche.

3) "Tricher" sur des fichiers specs existants

Une solution simple pour examiner un fichier SPEC existant est de télécharger le SRC.RPM d'un logiciel et d'en extraire le fichier *.spec.

Personnellement, j'utilise *mc* , une application "console" qui lit dans les paquetages RPM comme s'il s'agissait de tar.gz. Il est alors facile d'en extraire ou consulter un fichier texte quelconque.

4)Second essai

Après nos travaux communs d'apprentissage, mon fichier SPEC a évolué ainsi :

```
 _ %define name    bztarot
_ # Pour pas écrire 10 fois le titre dans le spec, on définit une variable "name"
_ # qui contient le nom du logiciel, ici c'est bztarot.
_ # idem avec les variables déclarées ci-dessous :
_ %define version 1.02
_ %define release 11mdk
_ %define section Amusement/Cards
_ %define title BzTarot
_ Name:           %{name}
_ Summary:        Jeu de tarot a 4 sous environnement graphique
_ Version:        %{version}
_ Release:        %{release}
_ License:        GPL
_ Group:          Games/Cards
_ URL:            http://www.beuselinck.com
_ Provides:       %{name} = %{version}-%{release}
_ Source0:        %name-%version.tar.bz2
_ Source1:        %name-icons.tar.bz2
_ Packager:       V.Beuselinck <vincent@beuselinck.com> _ BuildRoot:      %_tmppath/%name-buildroot
_ %description
_ Ce jeu de tarot vous permet de jouer seul contre l'ordinateur
 dans des parties a 4. Respecte la reglementation de la Federation
 Francaise de Tarot.
_ # Là, j'ai dû enlever les accents car pour rpm et urpmi ça va mais pour rpmdrake, ça n'affichait plus
rien.
_ %prep
_ %setup -q
_ %setup -q -T -D -a1 # unpack icons
_ #%patch1 -p0
```

```
_ # Dans cette section, on dit à RPM de décompresser les sources. Pour la première ligne, le -a0 de la source0 est implicite. Mais pour les autres sources, il faut numéroter...
_ %build
_ dcc tarot.dpr
_ # Dans cette section, on effectue la compilation en lançant le compilateur en ligne Kylix, à savoir dcc, sur le projet tarot.dpr
_ %install
_ rm -rf %buildroot
_ %__install -D -m 755 tarot %buildroot/%_bindir/%{name} _ # un petit nettoyage de "buildroot" puis on y installe le fichier compilé dans le virtuel /usr/bin
_ # Menu
_ mkdir -p %buildroot/%_menudir
_ cat > %buildroot/%_menudir/%name << EOF
_ ?package(%name): \
_ command="%_bindir/%name" \
_ needs="X11" \
_ icon="%name.png" \
_ section="%section" \
_ title="%title" \
_ longtitle="%Summary"
_ EOF
_ #Ci-dessus, on crée les entrées de menu pour les menus mandrake.
_ # icones et librairies
_ %__install -D -m 644 %{name}48.png %buildroot/%_liconsdir/%name.png
_ %__install -D -m 644 %{name}32.png %buildroot/%_iconsdir/%name.png
_ %__install -D -m 644 %{name}16.png %buildroot/%_miconsdir/%name.png
_ %__install -D -m 755 /usr/lib/libborqt* %buildroot/usr/lib/
_ # on installe également dans le système de fichiers virtuel les icones et librairies
_ %post
_ %{update_menus}
_ #ça, ça marche pas, mais j'ai déjà repéré des syntaxes différentes, j'irai à la pêche pour trouver la bonne syntaxe. Le but est de demander au gestionnaire de fenêtres d'actualiser son menu puisqu'on vient d'y mettre une nouvelle entrée. Sinon, il faut attendre le prochain boot pour que ça apparaisse.
_ %postun
_ %{clean_menus}
_ %files
_ %defattr(0755,root,root,0755)
_ %_bindir/*
_ /usr/lib/libborqt*
_ %defattr(0644,root,root,0755)
_ #%doc COPYING LICENSE README INSTALL Changelog AUTHORS
_ %_menudir/*
_ %_miconsdir/*
_ %_iconsdir/*
_ %_liconsdir/*
_ # Et là, on dit à RPM : tout ce que j'ai mis dans le système de fichier virtuel,  tu _l'installeras au même endroit mais dans le vrai système de fichier, quand quelqu'un te demandera d'installer le RPM
_ %changelog
_ * Mon Nov 13 2004 Vincent Beuselinck <vincent@beuselinck.com> 1.02-11mdk
_ - correctif d'une erreur en quittant le jeu
_ [..]
```

YAHooouuuuuuuuuuuuuu ! Voilà mon RPM est fait.

Prochaines étapes : séparer les librairies, franciser dans les règles de l'art le RPM, faire que la mise à jour des menus soit immédiate, mettre le compilateur Borland dcc comme « BuildRequires », c'est-à-dire nécessaire pour compiler les sources...

Post-scriptum :

Dernière partie : Accessoires et conclusion

Retour à la 3ème partie : La démarche

