http://clx.asso.fr/spip/?2-Les-principes-de-la-creation-d

2- Les principes de la création d'un RPM

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

Date de mise en ligne : dimanche 16 janvier 2005

Les principes

Pour comprendre comment construire un RPM, il est plus
simple de connaître d'abord les grandes lignes de la création d'un RPM...

RPM travaille proprement

Lorsque vous installez un logiciel à partir
de son code source, vous devez d'abord décompresser l'archive correspondante qui porte le plus souvent l'extension *.tar.gz* ou *.tar.bz2*. En effet, le code source est souvent
fourni sous la forme d'une archive compressé.
Il faut donc décompresser l'archive
dans un dossier préalablement choisi (ou bien copier l'archive dans un dossier particulier et l'y
décompresser, ce qui revient au même).
Puis, il faut préparer la compilation du programme en adaptant un fichier de configuration (*Makefile*) en fonction de votre architecture, des bibliothèques de fonctions présentes sur votre système, *etc.* Pour cela, vous utilisez la commande `./configure`. Vous lancez ensuite la compilation, via un programme spécifique â€"`make` â€" et enfin
l'installation dans les dossiers de destination
finale [1] est effectué via la commande *make install*.

Or, vous avez sûrement remarqué que ces opérations ne sont pas nécessaires avec un RPM. En effet, Il n'y a pas de
manipulation préalable à l'installation d'un RPM,
ni de commandes à lancer pour terminer
celle-ci. Le fichier SPEC élaboré lors de la construction du RPM s'occupe de réaliser toutes ces opérations. Tout comme *urpmi* installe automatiquement un RPM en téléchargeant et installant les dépendances nécessaires, etc.
Le principe de construction d'un RPM consiste donc à configurer un fichier de manière à ce que toutes ces opérations soient effectuées. C'est en cela que RPM fonctionne proprement.

Mais ce n'est pas le seul principe de "propreté"
du RPM. Lors de la création d'un RPM, là encore, le
principe de propreté prévaut. RPM va travailler dans un dossier que vous lui aurez
désigné et il faudra y créer plusieurs
sous-dossiers dans lesquels il rangera les
fichiers par catégorie : les sources, les specs,
les binaires (par architecture), les fichiers temporaires...

Et parmi ces repertoires, le dossier "*buildroot*" est
là pour servir de pseudo-racine du système. En
effet, s'il est nécessaire d'agir dans d'autres
dossiers lors de l'installation [2] (avant d'obtenir le RPM final, il faut faire quelques expérimentations), il est
inacceptable que RPM puisse agir sur le système d'exploitation
Linux et ainsi le rendre instable à cause d'une
expérience au résultat surprenant (suppressions ou
modifications de fichiers par exemple). C'est la
raison pour laquelle RPM utilise le buildroot :

une fausse arborescence racine dans laquelle il reproduira les dossiers nécessaires propres à Linux : `/usr/bin`, `/etc`, ...

Ainsi, si la fabrication du RPM n'a pas le résultat escompté, RPM ayant travaillé dans un dossier séparé, votre système Linux ne sera pas encombré par des fichiers inutiles ou dangereux.

RPM est une polka.

C'est-à-dire que l'on peut clairement distinguer deux temps dans sa façon de construire une archive.

Tout d'abord, RPM décompresse les fichiers sources à partir du dossier RPM/SOURCES. Attention, il faut prévoir que ces sources soient compressés dans un sous-dossier du nom (nom et version) de votre logiciel pour que la décompression se fasse dans RPM/SOURCES/logiciel_version. Par exemple, l'archive TAR de BzTarot contient un répertoire bztarot-1.02 qui contient les sources. C'est généralement le cas avec les sources que vous irez chercher sur le Net.

Ensuite, il va compiler le logiciel, utilisant pour cela votre dossier temporaire. Si la compilation se passe bien, le tout sera installé dans RPM/BUILD. Et pour finir cette première période, il va installer les fichiers dans le buildroot pour simuler l'installation. Les tests, vérifications et préparations sont terminées, le second temps est celui de la fabrication des RPM eux-même.

Il va créer le SRPM, un RPM particulier qui contiendra les sources, le spec, la description, bref tout ce qu'il faut pour quelqu'un qui voudrait recréer le RPM. Ce paquet n'est absolument pas exploitable, dans le sens où son installation ne permettra pas d'utiliser le logiciel. Enfin, il va créer le RPM qui contiendra tous les fichiers nécessaires au fonctionnement et à l'installation du logiciel, mais pas les éléments qui permettraient de refaire un RPM (pas de spec, ni source).

Certains estiment qu'il y a entre ses deux périodes, une période intermédiaire de débogage, correctifs, modifications, jurons et multiples

essais. C'est peut-être exact mais en scrutant aux rayons X les RPM et SRPM, cela ne se voit pas...

Post-scriptum :

3Úme partie : La démarche

Retour sur la 1Úre partie : Pourquoi créer ses propres paquetages

---

[1] `/usr/share/man` pour les pages man ; `/usr/bin` pour beaucoup de logiciels ; `/etc` pour les fichiers communs de configuration ; etc.

[2] Voir plus haut ; et note 1.