

APPIETTO Christophe
BOCQUET Joanes

Nice, le 20 Juin 2003

LICENCE INFORMATIQUE

TRAVAIL D'ETUDE ET DE RECHERCHE

**Étude comparative
des formats et
systèmes de
paquetage des
distributions Linux**

Université de Nice Sophia-Antipolis

Encadrant : M. Pierre CRESCENZO

SOMMAIRE

I)	INTRODUCTION.....	3
II)	Système de paquetage RPM.....	4
	- Historique	
	- Structure	
	- Avantages	
	- Inconvénients	
	- Outils Logiciels	
	- Aspects Philosophiques	
III)	Système de paquetage DEB.....	10
	- Historique	
	- Structure	
	- Avantages	
	- Inconvénients	
	- Outils Logiciels	
	- Aspects Philosophiques	
IV)	Système de paquetage SLP.....	13
	- Historique	
	- Structure	
	- Avantages	
	- Inconvénients	
	- Outils Logiciels	
	- Aspects Philosophiques	
V)	Système de paquetage TGZ.....	17
	- Historique	
	- Structure	
	- Avantages	
	- Inconvénients	
	- Outils Logiciels	
	- Aspects Philosophiques	
VI)	Outil Alien.....	18
VII)	Tableau Comparatif.....	20
VIII)	Synthèse.....	22
IX)	CONCLUSION.....	24

I) Introduction

Un paquetage est un ensemble de plusieurs fichiers de taille et de nombre variable permettant au concepteur d'un logiciel, d'une application ou autre, de distribuer facilement son produit. Chaque paquet possède une structure et une disposition des fichiers qui lui est propre, c'est ce qui détermine son format. Chaque distributeur sera libre de choisir un format pour la distribution de son produit mais également en fonction des propriétés attendues par le système de paquetage (conflits, dépendances, déplacement, mise à jour...). L'ensemble a pris le nom de distribution.

Lors du développement de linux, chaque système était préoccupé par son développement personnel, chacun a donc petit à petit simplifié sa démarche d'installation et développé son ou ses formats de paquetage sans consulter ses concurrents. Ceci a fait naître des systèmes de paquetages assez proches, basés sur les mêmes principes de facilité d'utilisation et de complexité dans la gestion des dépendances ou autres fonctions désormais indispensables.

Nous allons tenter de les différencier en détaillant leur structure d'une part mais également en exposant les différentes fonctions et possibilités que ces paquetages apportent à l'utilisateur et au système. Nous mettrons en avant leurs caractéristiques en exposant les avantages et inconvénients de chacun, ce qui nous conduira à une comparaison. Cette étude se déroulera en quatre parties distinctes qui sont en fait les quatre systèmes de paquetage étudiés (RPM, DEB, SLP, TGZ).



II) Système de paquetage RPM

➤ *Historique :*

A ses débuts, les problèmes de linux (Unix libre) étaient son manque de convivialité d'une part, mais surtout sa difficulté à l'installation et à la configuration qui demandait beaucoup de compétences et une certaine expérimentation pour être menée à bien.

Plusieurs distributions ont décidé de distribuer des cd d'installation contenant le système, c'est à dire les fichiers indispensables ainsi que les applications les plus utilisées et demandées en facilitant la vie des utilisateurs par des règles automatiques d'installations ou de dépendances lors de la sélection des composants à installer. RedHat, distributeur le plus connu, a voulu faire en sorte que son système soit bien évidemment simple à installer mais, il n'a pas voulu s'arrêter la ! Le RPM (RedHat Package Management) fait son arrivée aux alentours de 1996 ; la simplicité qui faisait le bonheur des clients linux lors de l'installation devient de la routine chez RedHat.

Très vite de nombreuses distributions se mettent à l'utiliser, RedHat (évidemment), Mandrake, Suse etc.... et il est choisi par la «Linux Standard Base» (qui est un organisme de standardisation des distributions Linux) pour la compatibilité des systèmes.

➤ *Structure d'un RPM :*

Il existe deux structures principales d'un paquet RPM : l'ancienne et la nouvelle (nommées ainsi dans un manuel RedHat).

L'ancienne version possédait un champ *lead* qui contenait les informations utilisées par le RPM lors d'extractions de fichiers ou pour lister les fichiers, qui semblait gêner pour l'extension des paquetages ; il ne sert donc maintenant que d'identification banale. Les anciennes versions sont incompatibles avec l'outil RPM actuel.

La structure actuelle est donc la suivante :

Extérieurement :

Le nom :

Le nom d'un paquet possède quatre champs distincts :

- Nom
- Version
- release (mise à jour de version)
- architecture (ex : i386)

Voici un petit tableau pour savoir à quoi correspondent les extensions :

*.i386.rpm	Archive binaire pour plate-forme Intel et compatibles
*.src.rpm	Archive contenant des sources toutes plates-formes
*.alpha.rpm	Archive binaire pour plate-forme DEC Alpha
*.ppc.rpm	Archive binaire pour plate-forme PowerPC

Intérieurement :

- Lead

Le champ lead ne sert plus qu'à l'identification du paquetage, pas à l'identification cryptée (qui sera le rôle de la Signature) mais une sorte de reconnaissance.

C'est donc une structure propre lead qui contient ces quelques informations :

- Nom
- Type (binaire, source)
- Architecture (ex : i386)
- etc.

Les informations réellement importantes se trouvent dans la structure Head qui suit.

- Signature

Ce champ représente la véritable identification, une authentification pas signature cryptée.

- Head

Toutes les informations du paquetage sont contenues dans une structure gigantesque possédant également des drapeaux, ce qui permettra une facilité et une rapidité d'accès. Nous allons citer quelques champs mais la liste des drapeaux qui suit parle d'elle-même, on y trouve tous les champs de la structure.

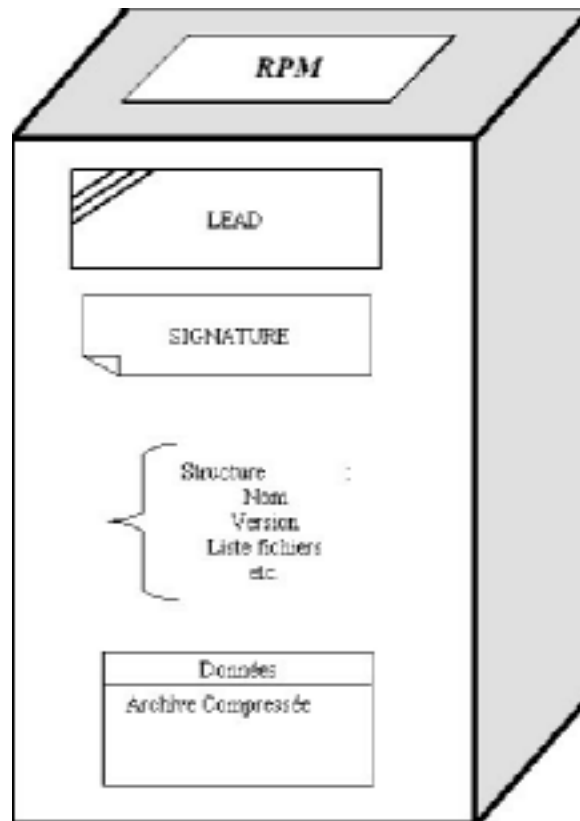
- Nombre magique désignant le début de l'en-tête
- Nom
- Version
- Liste des fichiers
- Liste des dépendances (drapeaux pour les versions)*
- etc.

*Lors de la création d'un RPM, les dépendances ou la compatibilité entre les différentes bibliothèques peuvent être ajoutées ; chaque paquetage pourra dépendre de plusieurs autres, et sera compatible avec certaines versions seulement d'un autre.

- Archives

Contient l'ensemble des fichiers les uns à la suite des autres et comprimés à l'aide de gzip.

Nous n'allons pas détailler l'archive, il s'agit d'une structure semblable à celle détaillée dans la partie qui concerne le système de paquetage TAR.



Drapeaux de la structure d'en tête

```
#define RPMTAG_NAME 1000
#define RPMTAG_VERSION 1001
#define RPMTAG_RELEASE 1002
#define RPMTAG_SERIAL 1003
#define RPMTAG_SUMMARY 1004
#define RPMTAG_DESCRIPTION 1005
#define RPMTAG_BUILDTIME 1006
#define RPMTAG_BUILDHOST 1007
#define RPMTAG_INSTALLTIME 1008
#define RPMTAG_SIZE 1009
#define RPMTAG_DISTRIBUTION 1010
#define RPMTAG_VENDOR 1011
#define RPMTAG_GIF 1012
#define RPMTAG_XPM 1013
#define RPMTAG_COPYRIGHT 1014
#define RPMTAG_PACKAGER 1015
#define RPMTAG_GROUP 1016
#define RPMTAG_CHANGELOG 1017
#define RPMTAG_SOURCE 1018
#define RPMTAG_PATCH 1019
#define RPMTAG_URL 1020
#define RPMTAG_OS 1021
#define RPMTAG_ARCH 1022
#define RPMTAG_PREIN 1023
#define RPMTAG_POSTIN 1024
#define RPMTAG_PREUN 1025
#define RPMTAG_POSTUN 1026
#define RPMTAG_FILENAMES 1027
#define RPMTAG_FILESIZES 1028
#define RPMTAG_FILESTATES 1029
#define RPMTAG_FILEMODES 1030
#define RPMTAG_FILEUIDS 1031
#define RPMTAG_FILEGIDS 1032
#define RPMTAG_FILERDEVS 1033
#define RPMTAG_FILEMTIMES 1034
#define RPMTAG_FILEMD5S 1035
#define RPMTAG_FILELINKTOS 1036
#define RPMTAG_FILEFLAGS 1037
#define RPMTAG_ROOT 1038
#define RPMTAG_FILEUSERNAME 1039
#define RPMTAG_FILEGROUPNAME 1040
#define RPMTAG_EXCLUDE 1041 /* Pas Utilisé */
#define RPMTAG_EXCLUSIVE 1042 /* Pas utilisé */
#define RPMTAG_ICON 1043
#define RPMTAG_SOURCERPM 1044
#define RPMTAG_FILEVERIFYFLAGS 1045
#define RPMTAG_ARCHIVESIZE 1046
#define RPMTAG_PROVIDES 1047
#define RPMTAG_REQUIREFLAGS 1048
#define RPMTAG_REQUIRENAME 1049
#define RPMTAG_REQUIREVERSION 1050
#define RPMTAG_NOSOURCE 1051
#define RPMTAG_NOPATCH 1052
#define RPMTAG_CONFLICTFLAGS 1053
#define RPMTAG_CONFLICTNAME 1054
#define RPMTAG_CONFLICTVERSION 1055
#define RPMTAG_DEFAULTPREFIX 1056
#define RPMTAG_BUILDRoot 1057
#define RPMTAG_INSTALLPREFIX 1058
#define RPMTAG_EXCLUDEARCH 1059
#define RPMTAG_EXCLUDEEOS 1060
#define RPMTAG_EXCLUSIVEARCH 1061
#define RPMTAG_EXCLUSIVEEOS 1062
#define RPMTAG_AUTOREQPROV 1063
#define RPMTAG_RPMVERSION 1064
#define RPMTAG_TRIGGERSCRIPTS 1065
#define RPMTAG_TRIGGERNAME 1066
#define RPMTAG_TRIGGERVERSION 1067
#define RPMTAG_TRIGGERFLAGS 1068
#define RPMTAG_TRIGGERINDEX 1069
#define RPMTAG_VERIFYSCRIPT 1079
```

➤ *Avantages :*

- L'outil RPM, développé pour le système RedHat à la base nous permet une grande facilité et simplicité d'utilisation.

Les dépendances avec les autres paquetages sont pratiquement gérées automatiquement, lors d'une installation un message nous demande si nous approuvons les choix de paquetages indispensables à notre installation et le reste est automatisé. Ceci est important car ce que beaucoup appellent de nos jours « le clique bouton » lors d'une installation d'un système RedHat se retrouvera lors de n'importe quelle installation de paquetages.

- Tout conflit avec un autre paquetage déjà installé sera également repéré et des mesures pourront être prises pour éviter tout dysfonctionnement, ou la suppression si celui-ci n'est plus utile (c'est l'utilisateur qui devra le signaler) ou une mise à jour si celle-ci est disponible et compatible.

- Des tâches peuvent être nécessaires avant et après l'installation, par exemple une sauvegarde ou la suppression de fichiers temporaires...

- Le paquetage gère également les fichiers de configuration (fichiers qui gardent les préférences, choix ou actions de l'utilisateur en mémoire). L'outil RPM vérifie si le fichier de configuration de l'application présente sur le disque a subi des modifications après son installation. Si c'est le cas, cela signifie que l'utilisateur y a fait des réglages ou autres. Ce fichier pourra être conservé.

Il est possible grâce à l'outil RPM d'installer un paquetage directement à partir d'une url ftp sans avoir à télécharger l'intégralité du paquet antérieurement.

Ex :

- `rpm -I ftp://ftp.truc.fr/emacs-20.7-4.rpm`

- Les sources des logiciels sont présentes dans le paquetage. Même en cas de modifications, de corrections ou d'extensions, les sources d'origines sont toujours présentes et isolées de toutes modifications qui se font sans changement des sources originales.

- Si un fichier a été effacé par mégarde, on a la possibilité de savoir s'il était utile et si c'est le cas quel paquetage l'utilisait. Tout ceci en vérifiant le système par une simple commande utilisant l'outil RPM :

- `rpm -Va`

Si pour un programme quelconque nous voulons avoir des informations, nous avons la possibilité de savoir de quel paquetage il provient, d'avoir une brève description et si nous le désirons, nous pouvons là encore avec une simple commande obtenir le lieu de la documentation fournit avec le paquetage dont il dépend :

- `rpm -qdf /usr/local/bin/free`

Voila le résultat fourni :

```
/usr/share/doc/procps-2.0.11/BUGS
/usr/share/doc/procps-2.0.11/NEWS
/usr/share/doc/procps-2.0.11/TODO
/usr/share/man/man1/free.1.gz
/usr/share/man/man1/oldps.1.gz
/usr/share/man/man1/pgrep.1.gz
/usr/share/man/man1/pkill.1.gz
/usr/share/man/man1/ps.1.gz
/usr/share/man/man1/skill.1.gz
/usr/share/man/man1/snice.1.gz
/usr/share/man/man1/tload.1.gz
/usr/share/man/man1/top.1.gz
/usr/share/man/man1/uptime.1.gz
/usr/share/man/man1/w.1.gz
/usr/share/man/man1/watch.1.gz
/usr/share/man/man5/sysctl.conf.5.gz
/usr/share/man/man8/sysctl.8.gz
/usr/share/man/man8/vmstat.8.gz
```

➤ *Inconvénients :*

- Lors de la gestion des conflits, il faut préciser que celle-ci ne se fait pas avec tous les programmes présents. En fait les conflits ne sont gérés que pour les applications issues de paquetages RPM, tout autre dont l'installation provient d'un autre système de paquetage sera ignorée et le conflit ne sera pas géré.

- Il est nécessaire d'utiliser l'outil RPM pour installer un paquetage du même nom. Nous n'avons pas la possibilité d'utiliser un outil standard ce qui va obliger la RedHat à fournir son outil RPM pour permettre l'installation sur certaines distributions. Ceci il faut le dire limitera l'utilisation du RPM aux grandes distributions compatibles.

- Lors d'une installation d'un paquetage à partir d'une url, il faut savoir que s'il y a des dépendances avec d'autres paquetages non présents en local ou sur un disque, ceux-ci ne pourront pas être installés.

- La création d'un rpm se fait en utilisant la commande
rpm -ba toto.spec

Cependant, vous devez être également l'auteur du fichier .spec, et définir dans celui-ci les principales propriétés de construction du RPM en plusieurs phases, il y en a 5 avec pas moins de 4 répertoires ou il va falloir aller et exécuter des commandes.

Tout ceci montre qu'il est difficile de créer un RPM ; l'outil simplifie donc la vie des utilisateurs mais ce n'est sûrement pas le cas pour les développeurs.

➤ *Outils et Logiciels*

- Outils Textes :
 - **RPM** (pour empaqueter, déempaqueter...)
 - Purp

- Outils Graphiques :
 - glint
 - gnorm (gnome)
 - kpackage (kde)
 - Software Manager (Mandrake)

➤ *Aspects philosophiques :*

Le RPM a été créé pour simplifier la vie des utilisateurs mais également celle des développeurs. En effet, le but était de permettre à un programmeur d'une application de pouvoir l'empaqueter avec une autre application d'un autre programmeur, indépendamment de la manière dont celle-ci a été empaquetée et configurée. De plus, toute modification d'une application sera séparée de celle-ci, dans une collection distincte ce qui permettra de conserver les sources intactes.

Il est également plaisant de disposer des sources d'une application que nous acquérons, pour une meilleure compréhension ou une éventuelle modification (*les applications étant pour la plupart libres « open source », chaque contribution permettra une amélioration*).

III) Système de paquetage DEB

➤ *Historique*

Dans les débuts de cette distribution, comme pour la plupart d'ailleurs, chaque paquet était composé d'une archive de sources que l'utilisateur devait décompacter (détarer en langage populaire) aux endroits adaptés. Très vite le projet prenant de l'importance, vers 1994 il a fallu améliorer le système d'installation, Ian Murdock développa le premier outil d'empaquetage dpkg permettant de créer des paquets dans un format binaire.

Environ un an après, Ian Jackson prit en charge le développement d'outils d'empaquetage, il développa dpkg-deb (se servant des travaux de son prédécesseur) puis une interface pour faciliter son utilisation dpkg. Petit à petit la gestion des dépendances, des conflits, la gestion de sécurité par signature cryptée s'intègrent à ce format qui devient le DEB.

Le système de paquetage DEB est moins vieux que le RPM mais il est né d'un outil basé sur le dpkg de Murdock qui est bien plus ancien que le premier outil assez simpliste de la RedHat.

➤ *Structure d'un DEB :*

On peut remarquer que la structure est une des plus simple (le tar mis à part) et des plus faciles d'utilisation.

• Données Binaires

Ce sont ce que l'on appelle généralement les méta données, les données relatives au paquetage.

Elles regroupent :

- Le nom du paquetage
- La version
- Les dépendances
- Une description (rôle du paquetage, dépendances principales)
- Et les scripts de maintenance

Les champs sont séparés pas « : » et les différents blocs par une nouvelle ligne.

• control.tar.gz

Contient les scripts de contrôle décrits précédemment, il est donc assez facile de les modifier ou de les changer sans devoir modifier les données du paquetage.

• data.tar.gz

Regroupe les données proprement dites, les fichiers à installer, les bibliothèques, etc.



Bocquet/Appietto

➤ *Avantages :*

- A l'inverse de RPM, pour utiliser les fonctionnalités d'une archive DEB nous utilisons un outil standard « dpkg ».

En voici quelques exemples :

- `dpkg -i toto.deb` (`--install`)

Installe le paquetage toto en faisant une copie (backup) de l'ancienne version.

- `dpkg -r toto.deb` (`--remove`)

Supprime le paquetage toto mais préserve le fichier de configuration.

- `dpkg -c toto.deb` (`--contents`)

Liste les fichiers contenus par le paquetage

- `dpkg --configure toto.deb`

Configure le paquetage toto.deb

- etc.

- Pour ne pas limiter DEB aux seules possibilités, formats et actions de dpkg, une application est présente, dpkg-deb qui permet d'enrichir le paquetage DEB en apportant différents formats d'archivage et compression.

- Les paquetages ne s'installent pas seulement en local, ils peuvent l'être (grâce à l'utilisation d'outils standards et non de « softwares managers ») par ftp, MS_Dos, NFS.... Nous pouvons remarquer que grâce à certains outils disponibles sur la distribution Debian, l'installation d'un paquetage est vraiment simplifiée ; Il n'est par exemple pas nécessaire comme pour le RPM de vérifier les dépendances lors d'un téléchargement car, pendant l'installation d'un paquetage, toute dépendance non trouvée sera téléchargée et installée automatiquement.

- Il est également facile d'exécuter des actions pré ou post installation (suppression) en ajoutant ou modifiant les scripts :

- preinst
- postinst
- prerm
- postrm

Il est de même très aisé de modifier les paramètres initialisés par défaut en changeant les scripts de maintenance:

- config
- tigger
- template

- Nous avons à notre disposition dselect, puissant outil utilisant dpkg, il va lister les paquets disponibles, notant les dépendances, les fichiers déjà installés avec les versions actuelles et disponibles sur Internet.. Il va permettre de gérer les quelques 8700 paquetages de la distribution debian.

Il permet :

- La sélection des paquetages pour l'installation avec une sélection automatique des dépendances.
- La sélection de paquetages pour la suppression avec (*c'est ici la puissance de cet outil*) la suppression des paquetages qui ne seront plus nécessaires et des avertissements pour les paquetages qui, dépendants de notre sélection, ne fonctionneront donc plus.
- La mise à jour de paquetages déjà présents.
- Configurer tout paquet qui ne l'est pas encore.

➤ *Inconvénients :*

- Les scripts souvent utilisés pour les installations posent de nombreuses questions auxquelles un utilisateur débutant ne sait pas toujours répondre.

- Un outil standard est utilisé pour ce paquetage mais, dpkg qui a des fonctions de base mères ou des applications comme apt-get, *libapt-pkg* sont nécessaires pour mieux gérer le DEB. La gestion des interdépendances, les mises à jour ou téléchargements de paquetages manquant automatiques (*moyens disponibles* : local, cdrom, http, ftp), ceci se fait par transparence mais l'outil dpkg ne suffit pas à lui seul.

- Seul un mode texte est disponible, rendant le produit peu convivial.

➤ *Outils et Logiciels :*

- *dselect* Gestion de paquets local et distants
- *dpkg* Outil principal d'installation (utilise par dselect)
- *apt-get* Autre outil gérant également les paquets distants (également utilisé par dselect)
- *tasksel* Gère les taches
- *aptitude* Installation de paquets (apt)
- *deity* Idem
- *synaptic*, *gsynaptic* — autres interfaces graphiques pour APT
- *Stow* (utilise apt-get)

- Outils pratiques
 - *apt-cache* recherche de paquets dans le cache
 - *dpkg-reconfigure* configuration de paquets déjà installés
 - *dpkg-source* gestion de paquets sources
 - *dpkg-buildpackage* Faciliter et automatiser la création de paquets

➤ *Aspects philosophiques :*

Cette distribution a été l'une des premières à avoir un système de paquetages binaires avec un outil pour son utilisation, par de nombreux travaux dont ceux de Murdock et Jackson, le vieil outil remodelé fournit un format de paquetage puissant. La structure est restée assez simple mais les fonctions développées aujourd'hui le place en tête de liste.



IV) Système de paquetage SLP

➤ *Historique :*

Le système de paquetage `slp` est en développement depuis 1998. La distribution Stampede est la première à l'utiliser. Ce format de paquetage a été conçu par Matthew S. Wood. De nombreuses distributions ont apporté leur contribution depuis, surtout pour le rendre le plus portable possible car c'était l'idée principale de ce projet. Ceci étant réalisé par l'utilisation d'outils déjà existants mais existant sur toutes les distributions (`tar` et `bzip2`).

➤ *Structure :*

La structure de ce format de paquetage fait parti des plus simples par sa représentation physique mais des plus complexes pour ce qu'elle va permettre de gérer.

Un paquetage SLP se divise en deux :

- Les archives compressées de taille variable
- Une structure de donnée de taille constante

Les archives sont donc constituées de tous les fichiers contenus par le paquet, disposés les uns à la suite des autres (donc la taille varie en fonction de la taille et du nombre de fichiers) et compressés avec un outil puissant qui est `bzip2`.

La structure appelée aussi en tête du fichier se trouve en réalité à la fin de celui-ci. Elle est de taille constante (3785 bytes de données) et comporte 19 champs. Nous allons citer et expliquer les 16 principaux.

- FilesToRetain

Fichiers qui ne devront jamais être écrasés en cas de version déjà présente (par exemple des fichiers de configuration)

- InstallRecommendations

Recommandations en ce qui concerne l'installation d'un paquetage (surtout en ce qui concerne les besoins du système)

0 -> Exigé

1 -> Recommandé

2 -> Facultatif

3 -> Non recommandé (souvent danger pour le système)

- DistributionMagicNumber

Champ initialisé pendant la manipulation des dépendances, il correspond à un identifiant de la distribution utilisée sur la machine.

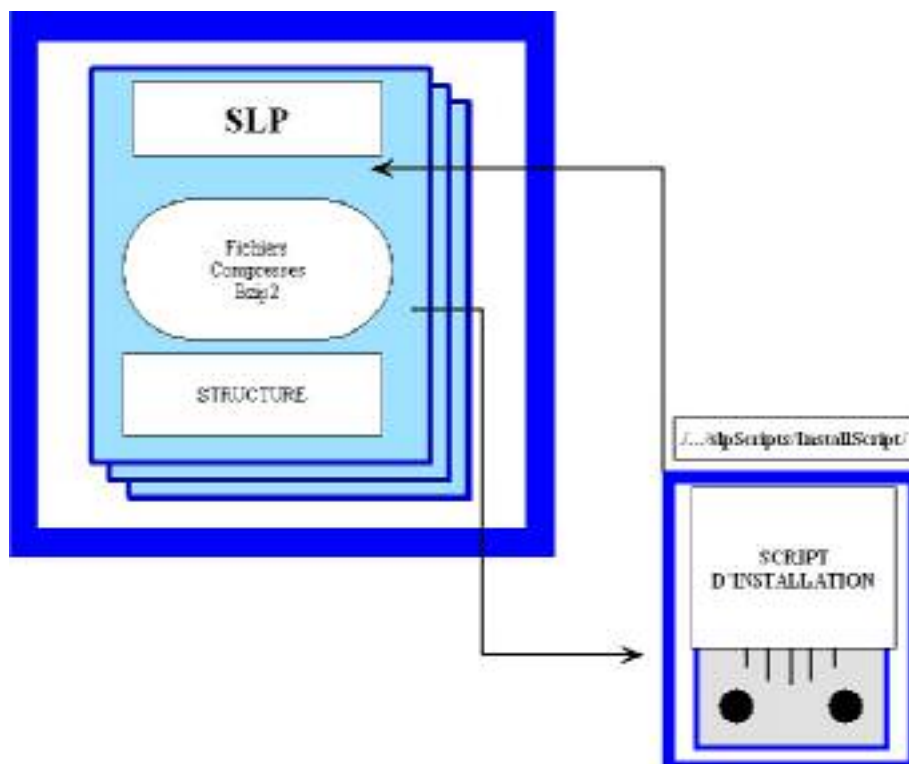
- PackageReleaseIndex

Au cas où un paquet était conçu pour une version spécifique de distribution, qui contient un morceau particulier de logiciel d'une révision en raison de la résolution de bug, ce champ serait incrémenté. Sinon sa valeur par défaut est 1.

- DistributionReleaseIndex

Désigne la version ou une correction particulière de la distribution qui est requise (ou plusieurs)

- PackageConflictsWith
Liste des paquetages avec lesquels il y aura des conflits.
- InstallScript
Contient le chemin « PATH » du script d'installation.
- DescriptionShort
Contient la description courte du paquetage, seuls les points essentiels sont mentionnés
- DescriptionLong
Contient la description complète du paquetage (du logiciel éventuellement)
- DependsRequired
Liste des dépendances requises pour le fonctionnement du paquetage.
- CryptographicSignature
Signature pour authentification de paquetage.
- PackageOrigine
Origine du paquetage et concepteur.
- PackageCreationDate
Date de création du paquetage.
- SoftwareVersion
Version de l'application.
- SoftwareName
Nom du logiciel (ou autre).
- AdvancedInstallScript
Ce champ a 0 ou 1 pour valeur, il indiquera au scripts d'installation si il devra effectuer certaines actions en pré/post installation (valeur 1) ou se contenter d'installer.



➤ *Avantages :*

- L'avantage principal de ce système de paquetage est ce pour quoi il a été conçu, une très grande portabilité. En effet, ne devant utiliser que des outils standards pour le

déempaquetage, celui-ci est possible quelle que soit la distribution utilisée, et rendant la distribution plus facile à utiliser.

- Les données du paquetage contiennent les informations relatives aux droits d'auteurs. Ceci est considéré comme un avantage lors de classements par auteurs par exemple de différentes applications.

- Les priorités sont également gérées, c'est-à-dire que lors d'installations de plusieurs paquetages pour des mises à jour, si des paquetages de priorité plus faibles sont considérés comme nécessaires, ceux de priorités supérieures seraient conseillés. Inversement ceci permet de ne pas installer des paquetages qui ne serviraient pas (priorité faible).

➤ *Inconvénients :*

- L'utilisation est décrite comme simple en ce qui concerne le décompactage mais, ceci n'est pas sans réserve, il est vrai que de simples outils standards suffisent mais dans ce cas l'en tête du fichier (que je rappelle se trouve à la fin) sera souvent ignorée. Un utilisateur voulant utiliser toutes les possibilités de ce système devra écrire un script d'installation, ce qui est complexe pour les utilisateurs peu avertis en programmation shell.

- De plus, la création de paquetage ne se fait pas à l'aide d'outils standards, ce qui une fois de plus destine ce paquetage à des utilisateurs confirmés ou du moins à des applications de fortes importances qui en utiliseront toutes les possibilités.

- Il n'y a pas de reconnaissance de ces paquetages par groupe, c'est-à-dire qu'ils ne pourront pas être classés dans une catégorie (par exemple bibliothèque). Ceci obligera à lister tous les fichiers si une bibliothèque (par exemple) est recherchée.

- Il n'y a pas de véritables programmes de pré ou post installation. Un script peut être exécuté pour l'installation mais on ne pourra pas faire ou désigner des programmes (ou encore scripts) de pré/post installation.

- Ce paquetage est le seul (de nos quatre étudiés) ayant des limites au codage dur, c'est-à-dire auquel des modifications ne pourront être appliquées comme une augmentation du nombre de caractères au nom ou l'ajout d'un champ dans la structure, etc. La structure qui est complexe (surtout comparée à celle du DEB) et ne permettra pas d'extensions futures sans redéfinition de son type, ce qui pourrait dans ce cas occasionner des problèmes de versions.

➤ *Outils et Logiciels :*

- Il est possible de n'utiliser que les outils standard (pas pour la création)
 - *tar*
 - *bzip2*
- Mais pour exploiter véritablement les possibilités fournies :

- *Guislp*
- *GPL*

➤ *Aspects philosophiques :*

Ce que nous pouvons dire sur ce système de paquetage est qu'il a été développé dans le but de trouver un format portable. Le but était d'avoir un système facile d'utilisation, possédant toute la puissance de n'importe quel autre système connu et utilisable partout. C'était le désir de la distribution Stampede qui n'a pas tout à fait atteint son but mais à développé un outil puissant.

V) Archive TAR/TGZ

➤ *Historique :*

Le système de paquetage tar est une commande système. Son origine remonte aux débuts de linux, il s'agit d'un archiveur ce qu'il y a de plus simple. Dans ses débuts, il servait à archiver les données sur bande magnétique d'où son nom tar (tape archiver).

➤ *Structure de l'archive :*

La structure du tar est une simple archive, il s'agit d'une entête de fichier contenant le nom, la taille et le chemin dans la hiérarchie de fichiers, c'est-à-dire le lieu ou le fichier sera détaré suivi du fichier désigné et ceci pour chaque fichier. L'archive TGZ est simplement une archive TAR normale qui a été compressée à l'aide de l'outil de compression gzip.

➤ *Avantages :*

- Le principal avantage de ce système de paquetage est qu'il suffit d'une seule ligne tapée dans un shell et seulement deux outils standards pour la créer.

- de même pour le déempaquetage, une seule ligne suffit. Ceci est idéal pour les petites applications ou pour sauvegarder les données.

Pour la sauvegarde de donnée, ce système de paquetage est ce qu'il y a de mieux, il conserve les permissions, les propriétaires, les dates, etc.

- De plus, il est possible d'exécuter un programme en post installation ce qui n'est pas le cas de tous les systèmes de paquetage.

- Et enfin, il est possible de savoir avec quelle version de l'outil le paquetage à été réalisé, ce qui est intéressant mais pas très utile car pour ce format, toutes les versions sont compatibles entre elles.

➤ *Inconvénients :*

L'archive ne peut pas être déplacée car l'extraction des fichiers se fait à partir de sa position hiérarchique dans le système de fichier.

De plus, aucune dépendance ou conflit n'est géré, ni même la présence de versions antérieures. Ceci est gênant car le décompactage se faisant à la position courante de l'archive, aucun renseignement sur le lieu ou pourrait se trouver une version antérieure ; pour vérifier son éventuelle présence, une recherche de fichiers est nécessaire et si celle-ci est fructueuse, il faut procéder à une suppression des fichiers avant l'installation de la nouvelle version.

La possibilité de faire un programme de post installation ou de configurer celle-ci n'est pas connue ; il est obligatoire de désinstaller les fichiers et de les copier si besoin est un à un dans les bons répertoires.

➤ *Aspects philosophiques :*

Ce système d'emballage qui est un outil standard donc disponible sur toutes les distributions linux, a été réalisé pour archiver n'importe quel ensemble de fichiers regroupés dans une hiérarchie propre aux systèmes linux. Il était au départ destiné à sauvegarder des données, applications, bibliothèques, etc. sur des bandes magnétiques. Son développement quasi immédiat a permis l'archivage dans des fichiers ainsi que la compression des données pour gagner de l'espace. Il est sur les distributions Linux actuelles un outil indispensable à tout utilisateur, du débutant au professionnel.

VI) Outil Alien :

Alien est un outil important que l'on ne peut ignorer, il permet de réaliser des conversions entre les différents types et formats de paquets linux, en particulier les quatre qui ont fait l'objet de notre étude.

Lors de conversions, il est possible que certaines données soient perdues comme des scripts de post-installation ou des dépendances si ceux-ci ne sont pas acceptés dans le nouveau format par exemple. Ainsi, l'archive sera moins performante que si elle était utilisée dans son format d'origine.

Alors pourquoi cet outil ?

Cet outil, qui est un script Perl, a été mis en place pour convertir un DEB en RPM ou en tgz... d'une manière aisée.

Il faut préciser que pour des conversions entre des systèmes de paquets puissants, leurs logiciels d'utilisation sont souvent requis. La conversion n'est pas parfaite, il est possible de perdre quelques dépendances ou scripts mais, il reste évident que par exemple lors de la conversion d'un RPM en tar on ne s'attend pas à trouver une archive de plusieurs GIGAS Octets contenant tous les paquets dont dépendait l'archive de départ. Cependant, on ne convertira pas un RPM utilisant plusieurs dizaines de bibliothèques et autres paquets en tgz.

Cet outil a surtout été développé pour les conversions entre DEB et RPM et s'est également très bien adapté au système slp, pour améliorer la portabilité des différents formats.

Voici un exemple pour comprendre la facilité de conversion entre paquetages

Convertir un rpm (clara-0.9.8-6.i386.rpm) en un tgz (dans ce cas en clara-0.9.8.tgz).

```
$ alien -t clara-0.9.8-6.i386.rpm
```

Warning: alien is not running as root!

Ownerships of files in the generated packages will probably be messed up.

clara-0.9.8.tgz generated

Clara ne possède pas de dépendances particulières et indispensables.

Conversion d'un deb en un rpm, en conservant les scripts et dépendances

```
# lien -r -c xfce_3.8.11-1_i386.deb
```

```
xfce-3.8.11-2.i386.rpm genette
```

On peut vérifier le paquetage en utilisant la commande rpm:

```
$ rpm -qiang xfce-3.8.11-2.i386.rpm
```

Voici la page du man de lien (lien --help)

lien --help

Usage: lien [options] file [...]

file [...] Package file or files to convert.

-d, --to-deb Generate a Debian deb package (default).

Enables the following options:

--patch= Specify patch file to use instead of automatically looking for patch in /var/lib/alien.

--nopatch Do not use patches.

--single Like --generate, but do not create .orig directory.

-r, --to-rpm Generate a RedHat rpm package.

--to-slp Generate a Stampede slp package.

-t, --to-tgz Generate a Slackware tgz package.

Enables the following option:

--description= Specify package description.

-i, --install Install generated package.

-g, --generate Unpack, but do not generate a new package.

-c, --scripts Include scripts in package.

-k, --keep-version Do not change version of generated package.

-h, --help Display this help message.

-v, --version Display alien's version number.

VII) Tableau comparatif :

Tableau réalisé par Joey Hess

<http://kitenet.net/~joey/pkg-comp/>

Tableau de comparaison des formats de paquetages.				
Caractéristiques	deb	rpm	tgz	slp
<i>Sécurité, authentification, et vérification</i>				
PGP- paquetages signés	non	oui	non	non
Décompte de fichiers, contrôles	oui	oui	non	non
Autorisations, permissions, propriétaires, etc.	oui	oui	oui	oui
Utilisable par les outils standard Linux				
Reconnaissable par fichier	oui	oui	non	non
Décompactable avec des outils standard	oui	non	oui	oui
Meta données accessible avec des outils standard	oui	non	N/A	non
Réalisable avec des outils standards	non	non	oui	non
Meta données				
Dépendances	oui	oui	non	oui
Recommandations	oui	non	non	non
Suggestions	oui	non	non	non
Conflits	oui	oui	non	oui
Paquetages virtuels et fournisseurs	oui	oui	non	??
Dépendances et conflits de versions	oui	oui	non	??
Dépendances booléennes complexes	oui	oui	non	non
Fichiers de dépendances	non	oui	non	non
Infos droits d'auteur	non	oui	non	oui
Répartition par groupes	oui	oui	non	non
Priorités	oui	non	non	oui
Fichiers particuliers				
Fichiers config	oui	oui	non	oui
Fichiers documentation	non	oui	non	non
Fichiers fantômes (reliquats)	non	oui	non	non
Programmes des paquetages				
Programmes binaires disponibles	oui	non	??	oui
Programme de pré installation	oui	oui	non	non
Programme de post-installation	oui	oui	oui	oui
Programme de pré destruction	oui	oui	non	non
Programme de vérification	non	oui	non	non

Déclencheurs (triggers)	non	oui	non	non
Possibilité d'extension, souplesse (scalability)				
Pas de limites d'encodage dur	oui	oui	oui	non
Nouvelles meta-données	oui	oui	N/A	non
Nouvelle section	oui	non	non	non
Version de format de données	oui	oui	non	oui

Dans le tableau de Joey Hess, nous remarquerons que les systèmes de paquetage DEB et RPM présentent beaucoup de similitudes et chacune des caractéristiques citées dans ce tableau sont au moins présente dans un de ces deux systèmes (mise à part bien sur la réalisation par outils standards).

Nous allons donc mettre en avant leurs différences :

Système de paquetage DEB :

- ❖ Est le seul décompressible avec des outils standards (disponible sur toute distribution)
- ❖ Permet également l'accès aux données par des outils standards
- ❖ Recommandation de paquetages dont la dépendance est presque certaine
- ❖ Suggestions de paquetages dont la dépendance est probable
- ❖ Gestion des priorités en ce qui concerne l'importance du paquetage pour le système.
- ❖ Flexibilité et amélioration possible de système, ajout de caractéristiques comme par exemple la signature PGP qui manque (pourquoi ?) sans altérer le format des archives.

Système de paquetage RPM :

- ❖ Les paquetages sont signés, ils contiennent une signature PGP permettant de vérifier l'identité du créateur.
- ❖ Gère les dépendances de fichiers. Un fichier ne sera installé que si un paquetage est présent ou un paquetage sera requis pour un fichier.
- ❖ Contient des informations sur les droits d'auteur.
- ❖ Possibilité d'une marque spéciale pour les fichiers de documentations, pouvant une recherche simple et rapide de documentations sur un paquetage.
- ❖ Possibilité de lancer un programme de vérification d'un paquetage installé.

En ce qui concerne le système SLP, le seul atout qu'il aurait pu avoir aurait été sa réalisation par des outils standards. Or ceci fut le cas à ses débuts mais n'est plus valable. Il présente toutes les qualités à la fois présentes simultanément pour les systèmes DEB et RPM (à part les programmes de pré installation et post-destruction).

Le tar est encore une fois mis à part, sa caractéristique prédominante reste sa réalisation très simple par un outil standard. Précisons qu'il est indispensable à tout système permettant l'utilisation de petites applications.

VIII) Synthèse

Les quatre formats de paquetages que nous venons de détailler sont les paquetages essentiels. La plupart des applications, outils ou logiciels populaires sont fournis dans ces formats de paquetages.

Le format TGZ que nous avons étudié et en dernière position est un peu à l'écart, destiné en fait à tout autre usage que les trois autres. En effet, lors de sa création tout ce qui est désigné comme dépendances, recommandations, suggestions, ou extension future n'était pas encore un « phénomène de mode » comme aujourd'hui. Il était aux débuts du projet GNU le seul utilisé et connu et est d'ailleurs aujourd'hui le seul présent sur toutes les distributions Linux. Il n'est plus employé maintenant que pour la sauvegarde des données plutôt que pour la distribution par son manque de fonctions. Il a un seul avantage par rapport aux autres formats de paquetages, c'est sa facilité d'utilisation. Malgré ses manques de performances comparés à ses concurrents dans cette étude, il est très utile pour sauvegarder ses propres données.

En résumé, il ne sera utilisé que pour l'archivage de données ou pour des applications de très faible importance ne possédant qu'un exécutable (ou quelques uns) et n'ayant pas besoin de bibliothèques particulières.

Nous avons beaucoup de caractéristiques appréciables dans un système tel que slp, qui est assez récent, où les concepteurs ont voulu faire cohabiter simplicité, portabilité et puissance d'utilisation. La facilité concerne le client, le désempaquetage d'une archive SLP ne nécessite que des outils standards, ce qui a permis à celle-ci d'atteindre son but en ce qui concerne cette facilité d'utilisation et sa portabilité sur toute distribution ou aucun logiciel ne sera nécessaire. Mais, en contre partie, la réalisation d'un paquetage s'avère assez complexe.

De plus, le désir d'associer à cette simplicité d'utilisation toutes les fonctions disponibles avec les autres systèmes de paquetages était un rêve difficilement réalisable. Beaucoup de fonctions ont été intégrées à ce système de paquetage, mais comme on a pu le remarquer dans le tableau de comparaison, toutes les caractéristiques dont dispose le SLP sont déjà présentes chez les deux autres formats étudiés, le DEB et le RPM, et il semble de plus que les fonctionnalités qui font actuellement les différences des deux systèmes de paquetages énoncés précédemment ne verront jamais le jour dans le format SLP.

L'absence d'amélioration prévue ne va pas augmenter sa popularité.

Les systèmes de paquetages DEB et RPM ne se différencient que très peu. Le choix de l'un d'entre eux ne dépendra que de la distribution visée par le programmeur, pour que son produit soit utilisable sur une distribution Debian il devra fournir un DEB et s'il veut que les autres grandes distributions l'utilise (Mandrake, Suze, ...) il devra fournir un RPM, ou autre ceci dépendra donc de la compatibilité des formats de paquetage.

Ces deux formats n'ont pas été dès leurs débuts en réelle compétition car, le RPM crée par RedHat fut un des premiers paquetages gérant dépendances, conflits, mises à jour ...; il à été mis en place assez tôt alors que le DEB était l'approfondissement et la complémentation d'un outil déjà présent et bien antérieur au RPM dans la distribution Debian.

Il semblerait que le format DEB soit un peu plus évolué car il permet ainsi que les mises à jour par mais, pouvons nous dire que ces caractéristiques feraient la différence pour un utilisateur expérimenté ?

Pour reparler de notre tableau de comparaison, on remarque en parcourant les deux premières colonnes qu'il y à 7 qualité du DEB non présente chez le RPM et exactement le même nombre de fonctionnalités disponibles seulement sur le RPM. Pour faire une comparaison plus approfondie ou plus exacte, il faudrait prendre ces différences point par point et évaluer leur nécessité.

Mais sommes nous aptes à la faire ?

Et si celles-ci ont été développées, ne répondent chacune à des besoins précis ?

IX) Conclusion

Les systèmes de paquetages sont très nombreux dans le monde Linux, on peut en distinguer au moins autant que de distributions. Il n'est pas possible de les classer selon leurs caractéristiques car chaque système répond à des besoins précis. Un distributeur devra donc bien réfléchir aux caractéristiques attendues lors de l'installation de son produit et bien entendu aux distributions visées (car chacune tolère un nombre fini de formats de paquetages) avant de faire son choix pour emballer son produit avant la distribution sur le marché, surtout entre DEB et RPM, issus d'origines différentes, mais ayant chacun leur points forts et faibles.

Sources :

Internet :

Tableaux Comparatifs

<http://kitenet.net/~joey/pkg-comp/>
<http://aadelmar.free.fr/comparatif-formats-paquetages.html>

DEB

<http://www.debian.org/releases/slink/i386/dselect-beginner.en.html>
<http://www.debian.org/doc/manuals/quick-reference/quick-reference.fr.html>
<http://www.linux-france.org/article/>

RPM

<http://www.linux-france.org/article/>

TGZ

<http://www.linux-france.org/article/>

autre...

<http://www-106.ibm.com/developerworks/linux/library/l-stow/?ca=dgrlnxw01STOW>
<http://linuxiens.tuxfamily.org/articles/divers/paquetages.php>
<http://kitenet.net/programs/alien/> (Alien)

Manuels :

deselect
dpkg
tar
Alien

Mailing Lists :

Debian
Mandrake
RedHat
Stampede

Livres :

UNIX - Le Tout en Poche Dave Taylor *Ed Campus Press*
Kit de Démarrage Linux RedHat