Spécification technique de réalisation

Corentin GUILLEVIC et Sarah HADBI

April 7, 2015

Contents

1 Introduction

Dans ce document nous nous intéresserons à la partie technique du cahier des charges, la STR (Spécification technique de réalisation), qui vise à fournir au prestataire ou au maître d'œuvre (MOE) de notre projet informatique le maximum de spécifications techniques que nous souhaitons valider. Cette partie nécessite un niveau d'expertise informatique non-négligeable et vise à assurer la pérennité du projet. En outre, dans le cadre de notre projet Paquito, nous :

- Préciserons les différences et les points communs entre les systèmes de fichiers (Debian, ArchLinux et Redhat)
- Effectuerons des modifications au sein de certains fichiers déjà présents
- Apporterons des solutions aux problèmes rencontrés (relatés dans le cahier des charges)

2 Systèmes de paquets

2.1 Points Communs

Chacune des distribution dispose d'un fichier spécial propre destiné à spécifier les méta-données du paquet ainsi qu'éventuellement définir sa procédure de construction. Le volume de données est variable selon le système de paquet :

- **Debian**: Le fichier **control** contient les informations sur le paquet
- **Archlinux** : Le fichier **PKGBUILD** contient les informations sur le paquet et une fonction appelée **build()** pour construire le paquet
- **RPM** : Le fichier **SPEC** contient les informations sur le paquet ainsi que plusieurs sections pour la construction du paquet

Les autres points communs entre les systèmes de fichiers sont :

- Les méta-données des les fichiers sus-cités sont globalement similaires : nom du paquet, version, dépendances, licence, etc...
- Les paquets sont organisés pour contenir dans une partie dédiée les fichiers du programme (par exemple, le répertoire **/usr/bin** est censé contenir l'exécutable).

- Les éléments communs du nommage des paquets pour les 3 systèmes comprennent le nom du paquet et sa version (tous deux définis dans le fichier spécial).
- Chaque système de paquet prévoit une commande spécifique propre pour initier la création du paquet :
  - **Debian** : dpkg-deb –build package_name
  - **Archlinux** : makepkg
  - **Redhat** : rpmbuild -ba rpmbuild/SPECS/package_name.spec
- Les trois distributions possèdent une variable contenant les dépendances du programme (dépendances à l'exécution). Cette variable est située dans le fichier spécial de chaque distribution :
  - **Debian** : *Depends:tcc>=1.4*
  - **Archlinux** : *Depends=('tcc>=1.4')*
  - **Redhat** : *tcc*

  Remarque : L'exemple est tiré de la dépendance à l'exécution "Tiny C" du programme HELLOWORLD.

2.2 Différences

2.2.1 Arborescence

Puisque chaque distribution impose une arborescence spécifique pour créer un paquet, cette partie vise à montrer et expliquer ces différences.

Debian Le répertoire du paquet sera constitué de deux parties :

- Une partie **control** (fichier **control** qui est dans le répertoire **DEBIAN**), qui contiendra les informations sur le paquet (version, nom, dépendances...)
- Une partie **data**, qui contiendra les fichiers du programme, tel que par exemple le répertoire **/usr/bin**, qui contiendra le ou les exécutables

Remarque : Le fichier **control** est propre à la distribution Debian

Archlinux Le répertoire du paquet contiendra uniquement la partie **data**, qui héberge les différents fichiers du programme. Par exemple : **/usr/bin**, **/usr/lib** pour les librairies, etc...

```
package/
├── DEBIAN/
│   └── control
└── usr/
    └── bin/
```

```
pkgbuild
└── package/
    └── usr/
        └── bin/
```

Redhat Il n'y a pas de répertoire spécial pour l'arborescence du paquet. À la place, il y a un répertoire **rpmbuild** qui contiendra les répertoires suivants :

SOURCES Contient l'archive contenant les sources du programme

BUILD Contient l'archive décompressée (celle présente dans le répertoire **SOURCES**)

RPMS Contient le paquet RPM binaire

SRPMS Contient le paquet RPM source

SPECS Contient le fichier **SPEC** qui est un fichier propre à RPM, il contiendra les informations sur le paquet ainsi que différentes sections

Remarque : La commande **rpmdev-setuptree** permet de simplifier la procédure en créant automatiquement l'arborescence suscitée.

```
rpmbuild/
├── SOURCES/
├── BUILD/
├── SPECS/
│   └── SPEC/
├── RPMS/
│   └── Paquet binaire
└── SRPMS/
    └── Paquet source
```

2.2.2 Construction du paquet

Debian La création du paquet se fait à partir d'un simple script shell. Il contiendra :

- Les commandes permettant la création de l'arborescence décrite précédemment
- Les commandes de compilation afin de compiler les sources et générer le ou les différents exécutables
- La commande permettant la création du paquet Debian : **sudo dpkg-deb –build package_name** (ou **package_name** est le nom du répertoire contenant l'arborescence du paquet)

En exécutant la commande **ls** sur le répertoire où se situe le script, nous constaterons la présence d'un répertoire (qui sera le répertoire décrit précédemment contenant l'arborescence du paquet) et du paquet Debian fraîchement créé.

Archlinux La création du paquet se fait à l'aide de la fonction **buid()** contenu dans le fichier **PKGBUILD** (qui est propre à la distribution Archlinux).

Rappel : Le fichier **PKGBUILD** contient les information sur le paquet (comme le fichier **control**) mais aussi la fonction **build()** qui gère la création du paquet.

En général, l'ordre des commandes dans la fonction **build()** est le suivant :

1. Déplacement dans le répertoire contenant les sources téléchargées
2. Compilation les sources
3. Déplacement dans le répertoire destiné à être le paquet

Une fois le fichier **PKGBUILD** rempli pour créer le paquet, il faut exécuter la commande **makepkg** (au niveau du répertoire qui contient ce fichier), ce qui aura pour effet de créer le paquet Archlinux. Du coup, on pourra l'apercevoir avec la commande **ls**, en plus du fichier **PKGBUILD**, et du répertoire qui contient l'arborescence du paquet.

Redhat La construction du paquet RPM est gérée par le fichier **SPEC**. Il contient les informations sur le paquet (à l'instar du fichier **control**) mais surtout des sections permettant d'arriver à la création du paquet? Par exemple :

- *%build* est une section du fichier **SPEC** qui permettra la compilation des sources
- *%install* est une section qui se chargera de déplacer les fichiers sources et exécutables dans les bons répertoires

Une fois le fichier **SPEC** rempli, on se place au niveau du répertoire qui contiendra le répertoire **rpmbuild** décrit précédemment, pour exécuter la commande **rpmbuild -ba rpmbuild/SPECS/package_name.spec Remarque :** Le nom du fichier **SPEC** est composé du nom souhaité pour le paquet suivi de l'extension **.spec**.

Il y aura deux paquets créés : un paquet binaire qui sera contenu dans le répertoire **RPMS** et le paquet source qui sera contenu dans le répertoire **SRPMS**.

Remarque : Le paquet source permet en l'installant de récupérer le fichier **SPEC** et les sources du programme ayant servis pour la construction du RPM associé.

2.2.3 Nommage

Une fois le paquet créé, il aura selon la distribution les noms suivants :

- **Debian** : nom-du-paquet_version-1_architecture.deb
- **Archlinux** : nom-du-paquet-version.tar.gz
- **Redhat** : nom-du-paquet-version-1.fc10.architecture.rpm

De plus, les noms des variables (méta-données) ne sont pas toujours les mêmes selon la distribution. Si on considère pour l'exemple la variable utilisée pour nommer le paquet, son nom n'est pas identique :

- **Debian** : *Package*
- **Archlinux** : *pkgname*
- **Redhat** : *Name*

2.2.4 Dépendances

En plus des dépendances citées précédemment (aux points communs), le système de paquet d'Archlinux prend en charge deux autres types de dépendances :

optdepends Paquet optionnel qui rajoute en général des fonctionnalités au programme

makedepends Dépendances exigées pour la compilation du programme

Le système de paquet RPM possède lui-aussi le type de dépendances exigées pour la compilation du programme (**BuildRequires**).

3 Retour sur l'existant

Cette partie est un résumé des points importants du projet Paquito 2014.

3.1 buildeb

Le fichier **buildeb** est un script shell qui permet de faciliter la création d'un paquet Debian à partir de sources. C'est en quelque sorte l'exécutable du projet Paquito 2014.

Pour concevoir le paquet Debian désiré, il faut remplir et compléter le fichier **build_config.yml** (qui contiendra toutes les méta-données). Il sera ensuite interprété par le script **buildeb** pour réaliser un paquet ou plusieurs paquets. L'exécution du script est divisée en 9 étapes :

1. Interprétation des options fournis en ligne de commande
2. Lecture du fichier de configuration (c'est-à-dire **build_config.yml** grâce au script **config_parser.pl**)
3. Recherche du **Makefile**
4. Création de logs (2 journaux, le contenu de l'un est la sortie standard des différentes commandes, l'autre contient la sortie de **debuild** [l'outil qui construit les paquets])
5. Tâches avant les opérations de construction (**BEFOREBUILD**)
6. Création des Tarballs (c'est-à-dire les archives qui contiennent le code source) (seulement pour les paquets contenant des librairies ou des paquets binaires)
7. Gestion du fichier copyright (il doit être présent pour répondre à la politique de Debian)
8. Construction des paquets :

(a) Création des fichiers Debian (avec **dh_make**)

(b) Modification des fichiers Debian

(c) Création du paquet source

(d) Construction du paquet

9. Lancement après les opérations de construction

3.2 config-parser.pl

Interprète le fichier **build_config.yml** pour y récupérer les informations. Pour cela, il doit impérativement se trouver dans le même répertoire que le script **buildeb**.

3.3 correct-lintian.pl

Détecte les éventuelles erreurs et vérifie la conformité du paquet Debian par rapport à la politique de Debian. Bien que ce script ne soit pas nécessaire pour la procédure de création de paquets, il est hautement recommandé de l'utiliser afin d'obtenir des paquets correctement formés.

Naturellement, il doit se trouver dans le même répertoire que le script **buildeb**.

3.4 build-config.yml

Le fichier **build_config.yml** (qui devra être renommé **paquito.yml**) est le fichier de configuration que devra remplir le développeur du logiciel pour former le paquet. Il est initialement constitué des champs suivants :

- **BINPACKAGENAME** : Nom souhaité pour le paquet binaire (si le paquet à construire n'est pas un binaire, laisser vide). Voir **PACKAGETYPE**
- **LIBPACKAGENAME** : Nom souhaité pour le paquet de librairies (si le paquet à construire ne contient pas de librairies, laisser vide). Voir **PACKAGETYPE**
- **INDPACKAGENAME** : Nom souhaité pour le paquet contenant des fichiers sans architecture (si le paquet à construire est destiné à une architecture, laisser vide). Voir **PACKAGETYPE**

- **VERSION** : Version du programme. Doit être incrémenté lors du remplissage du fichier de configuration pour que le paquet créé représente la dernière version disponible. Si vide, utilise la date (timestamp)
- **COPYRIGHT** : Copyrigth. Une liste de copyright est disponible pour faire un choix. Mais on peut spécifer à la place un fichier. Enfin, il est possible de laisser ce champ vide (le fichier **COPYING** ou **LICENSE** sera alors recherché)
- **DEVS** : Développeur(s) du projet. La mise en forme doit être "- *201X-201X Nom address@mail*". Si vide, recherche le fichier "AUTHORS" (qui doit se trouver dans le même répertoire que le **Makefile**)
- **PACKAGETYPE** : Type de paquet, dont il existe trois options : "s" pour un paquet binaire, "l" pour une librairie, et "s l" pour un paquet "hybride"
- **BUILDDEPENDS** : Liste de dépendances nécessaires à la construction du paquet (plus précisément lorsque Make est lancé, donc pour la compilation du logiciel). Il est possible, pour chaque paquet, de spécifier la version minimum requise
- **BINRUNDEPENDS** : Dépendances (à l'exécution) dans le cas d'un paquet binaire
- **LIBRUNDEPENDS** : Dépendances (à l'exécution) dans le cas d'un paquet de librairies
- **INDRUNDEPENDS** : Dépendances (à l'exécution) dans le cas d'un paquet sans architecture
- **BEFOREBUILD** : Liste des commandes à lancer avant le début de la construction du paquet. Attention : si une variable d'environment est changée ici, elle doit être rétablie par l'intermédiaire de **AFTERBUILD**
- **AFTERBUILD** : Liste des commandes à lancer après la construction du paquet
- **CONFIGUREFLAGS** : Options que l'on souhaite passer à **./configure**
- **BINARYNAMES** : Liste des binaires qui seront inclus dans le paquet binaire. Chaque entrée se présente sous la forme : "chemin du fichier à

inclure"::"chemin relatif (depuis /) vers le répertoire où devra se trouver après installation le binaire"

- **LIBNAMES** : Liste des librairies qui seront inclues dans le paquet de librairies. La syntaxe est la même que **BINARYNAMES**
- **HEADERNAMES** : Liste de fichiers d'entête (headers) qui seront inclus dans le paquet de développement. La syntaxe est la même que **BINARYNAMES**
- **INDNAMES** : Liste des fichiers sans architecture qui seront inclus dans le paquet sans architecture. La syntaxe est la même que BINARYNAMES
- **MANPAGES** : Liste des pages du manuel qui doivent être inclues dans le paquet binaire. Mettre seulement les noms, les dossiers d'installation ne sont pas demandés
- **DISTRIBUTION** : Distribution (et sa version) du paquet
- **BINPACKAGEDESCFILE** : Chemin vers le fichier de description pour le paquet binaire. Le fichier contient la descripteur que l'utilisateur peut consulter avec **dpkg -I** ou un gestionnaire de paquet graphique
- **LIBPACKAGEDESCFILE** : Chemin vers le fichier de description pour le paquet de libraires. Même chose que **BINPACKAGEDESCFILE**
- **INDPACKAGEDESCFILE** : Chemin vers le fichier de description pour le paquet sans-architecture. Même chose que **BINPACKAGEDESCFILE**
- **DEBFULLNAME** : Le nom du mainteneur du paquet
- **DEBEMAIL** : Adresse mail du mainteneur du projet
- **HOMEPAGE** : URL associée au paquet