

Création d'un paquet debian (.deb)

Remarque : La méthode détaillée ci-dessous utilise l'outil debhelper, il s'agit d'un outil mettant à disposition de nombreux scripts permettant d'automatiser et donc de faciliter l'empaquetage (comme par exemple la commande `dh_make` que l'on utilisera ci-dessous).

Exemple choisi pour cette documentation : empaquetage de MORSE, mais aussi TOM pour les explications sur les fichiers « install » et « postinst ».

1°) Paquets nécessaires pour la création de paquets debian

- build-essential
- cdb
- debhelper
- debootstrap
- devscripts
- dh-make
- fakeroot
- pbuilder

Commande : `sudo apt-get install build-essential cdb debhelper debootstrap devscripts dh-make fakeroot lintian pbuilder`

2°) Récupération des sources du logiciel à empaqueter

!! Attention : les sources du logiciel ne doivent jamais être modifiées directement.!!

- Si nécessaire, remettez l'archive contenant les sources au format `.tar.gz` (extraire les fichiers puis compresser à nouveau dans le bon format) :
`laas-morse-0.3-0-g7e139e9.tar.gz`
- Renommez l'archive sous la forme
`<nom_du_paquet>_<numéro_de_version>.orig.tar.gz`, veillez à simplifier le nom du paquet en un seul mot:
`morse_0.3.orig.tar.gz`
- Extrayez les fichiers sources afin d'obtenir un dossier sous la forme `<nom du paquet>-<numéro de version>` :
`morse-0.3`

Distinguez bien le tiret simple « - » et le tiret bas « _ ».

3°) Création du dossier « debian »

Ce dossier contient les fichiers nécessaires à la création d'un paquet, il se situera dans le dossier que vous venez d'extraire.

Pour le créer :

- Faire « `cd <dossier source>` »
- Puis appeler la commande « `dh_make` » de cette manière :
`<dh_make -s -e <votre_adresse_mail>`
Le paramètre « `-s` » permet de préciser directement que l'on souhaite créer un paquet avec un seul binaire, si vous souhaitez accéder aux autres types de paquets, supprimez-le de cette ligne de commande

- Valider avec la touche « entrée »

```

clement@clement-K50IN: ~/Stage/boulot/creationPaquet/finalPackages/morse/
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
clement@clement-K50IN:~/Stage/boulot/creationPaquet/finalPackages/morse/morse/mo
rse-0.3$ dh_make -s -e clement.garcia14@gmail.com
Maintainer name : clement
Email-Address   : clement.garcia14@gmail.com
Date            : Fri, 22 Apr 2011 15:28:49 +0700
Package Name    : morse
Version         : 0.3
License         : blank
Using dpatch    : no
Type of Package : Single
Hit <enter> to confirm:
Skipping creating ../morse_0.3.orig.tar.gz because it already exists
You already have a debian/ subdirectory in the source tree.
dh_make will not try to overwrite anything.
clement@clement-K50IN:~/Stage/boulot/creationPaquet/finalPackages/morse/morse/mo
rse-0.3$

```

Le dossier « debian » est alors créé, et contient beaucoup de fichiers à cause des exemples. Il va donc falloir le nettoyer avec la commande :

```
rm *.ex *.EX README* docs info
```

Normalement, il ne reste plus que 5 fichiers et un dossier nommé « source » qui n'a pas besoin d'être modifié. Liste de ces fichiers :

- changelog : détails des évolutions du paquet
 - nom_du_paquet_source (version_du_paquet)
 - descriptif des changements (changelog en soit)
 - nom de l'empaqueteur, jour, date heure timezone
- control : description du paquet source et du paquet binaire résultant
- copyright : copyright, licence, liste des contributeurs, etc
- rules : fichier exécutable
 - piloter la compilation lors de la création du fichier binaire (règles nécessaires à la compilation du paquet) : un makefile
 - à modifier seulement en cas de problème dans la compilation ou pour ajouter éléments particuliers à l'installation (script shell ou une page man par exemple)
- compat : indiquer la compatibilité debhelper (devrait contenir le numéro 7 ou plus, ne pas modifier)

Pour éditer correctement l'ensemble de ces fichiers, il est conseillé de voir tous les détails sur ce lien provenant de la documentation ubuntu et présentant le cas de chacun d'eux :

- http://doc.ubuntu-fr.org/tutoriel/creer_un_paquet
- mais aussi ce tutoriel vidéo pour plus de précisions :

1ère partie : <http://www.youtube.com/watch?v=zKLabXTqMc>

2ème partie : <http://www.youtube.com/watch?v=SwTp1YnehoI&feature=relmfu>

Conseil pour les informations du fichier « copyright »:

- pour la ligne « Upstream Authors » : consulter le fichier AUTHORS s'il existe, sinon lire le README ou chercher l'information sur le site de l'éditeur
- pour la ligne « Copyright » : regarder les headers des éventuels fichiers en C avec la commande → `find . -name '*.c' | xargs head | less` et/ou s'informer auprès des fichiers AUTHORS ou licence
- pour la ligne « License » : regarder le fichier COPYING et/ou licence

4°) Création d'une clé GPG

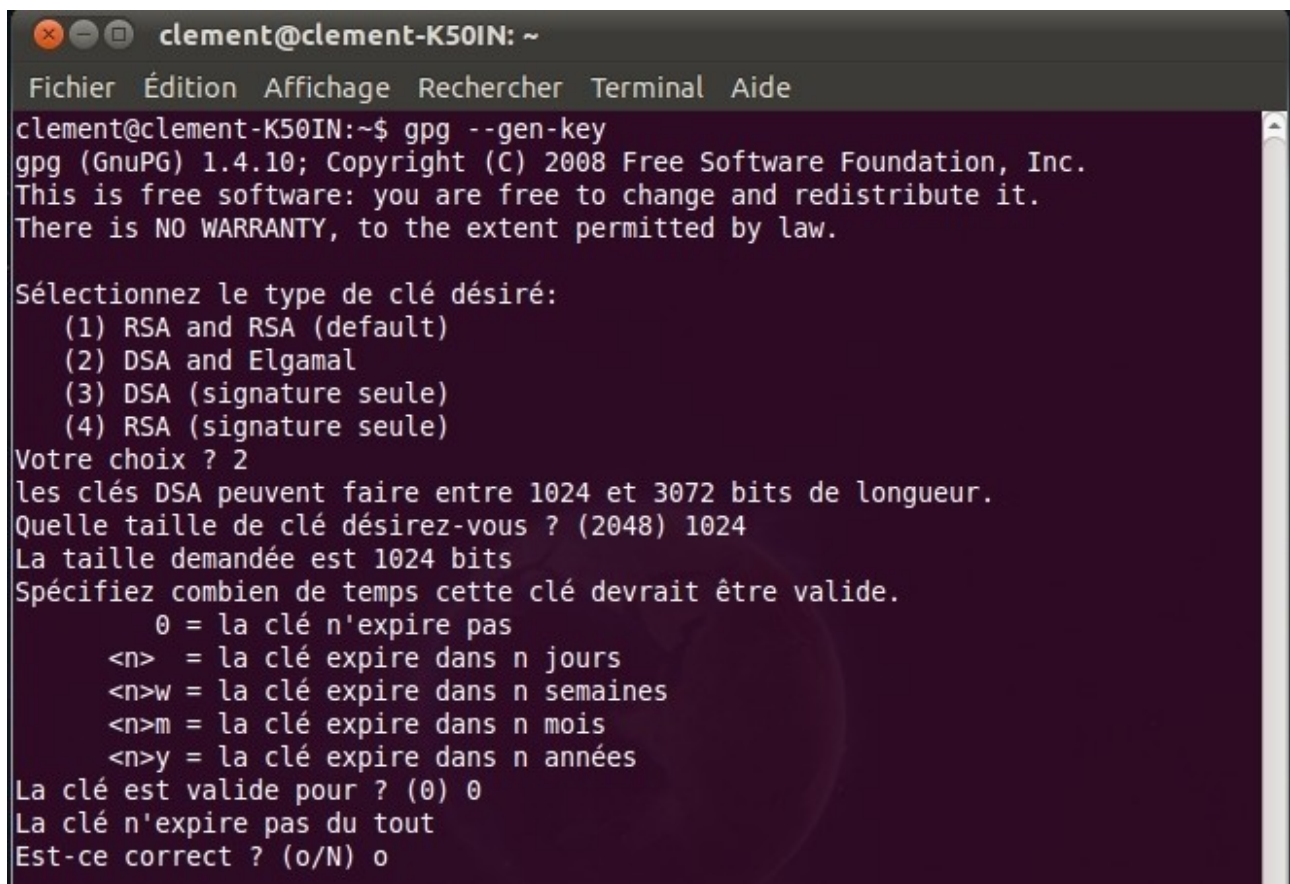
Une fois ces différents fichiers édités, avant même de reconstruire le paquet, il va falloir créer une clé GPG (GNU Privacy Guard) afin de le signer.

Commande pour créer une clé :

```
gpg --gen-key
```

A la question du type de clé, répondez « (2) DSA and Elgamal ».

Pour la taille, une clé de taille 1024 suffira. Choisissez la durée de validité, répondez au reste des informations et vous obtiendrez alors une clé GPG.



```
clement@clement-K50IN: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
clement@clement-K50IN:~$ gpg --gen-key
gpg (GnuPG) 1.4.10; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

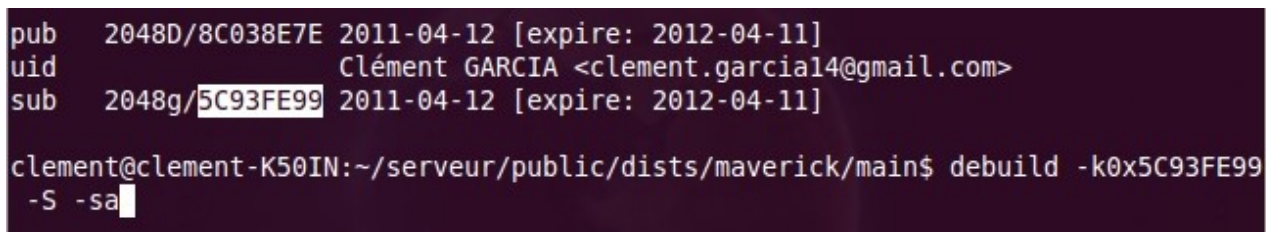
Sélectionnez le type de clé désiré:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (signature seule)
  (4) RSA (signature seule)
Votre choix ? 2
les clés DSA peuvent faire entre 1024 et 3072 bits de longueur.
Quelle taille de clé désirez-vous ? (2048) 1024
La taille demandée est 1024 bits
Spécifiez combien de temps cette clé devrait être valide.
  0 = la clé n'expire pas
  <n> = la clé expire dans n jours
  <n>w = la clé expire dans n semaines
  <n>m = la clé expire dans n mois
  <n>y = la clé expire dans n années
La clé est valide pour ? (0) 0
La clé n'expire pas du tout
Est-ce correct ? (o/N) o
```

Pour lister les clés GPG existantes et accéder à leurs identifiants : `gpg --list-keys`

5°) Construction du paquet

Placez vous à la racine du paquet et passons à présent à la reconstruction de celui-ci :
`debuild -k0x[iud de votre clé gpg] -S -sa`

Le -S permet de construire le paquet source.



```
pub 2048D/8C038E7E 2011-04-12 [expire: 2012-04-11]
uid Clément GARCIA <clement.garcia14@gmail.com>
sub 2048g/5C93FE99 2011-04-12 [expire: 2012-04-11]

clement@clement-K50IN:~/serveur/public/dists/maverick/main$ debuild -k0x5C93FE99
-S -sa
```

L'identifiant de la clé est surlignée dans l'image ci-dessus

Vous observerez alors que les fichiers .dsc, .build, .diff, et .changes ont été générés.

6°) Compilation et création du paquet binaire

Outil utilisé pour cette dernière étape : pbuilder.

Si le paquet possède un MakeFile, vous allez pouvoir directement passer à l'étape de pbuilder, sinon, étant donné qu'aucun fichier ne donne les règles de compilation, vous allez devoir créer un fichier nommé « install » dans le dossier « debian ». Le fichier se présente sous cette forme :



```
install ✕
1 |lib/* lib
2 |bin/* bin
3 |share/* share
```

Comme vous pouvez l'observer, il faut préciser la source du fichier dans le paquet à gauche, puis sur la même ligne et séparé d'un espace, la destination sur le disque dur.

Pbuilder permet de réaliser un chroot de votre système, l'intérêt est de pouvoir construire le paquet sans modifier votre environnement mais aussi de vérifier que les dépendances soient bonnes. Pour initialiser pbuilder, faites : `sudo pbuilder create`. Cette étape risque de prendre du temps, vous pouvez donc la lancer en début de travail sur un autre terminal.

Une fois l'outil pbuilder préparé, il ne reste plus qu'à compiler le paquet. En se plaçant au niveau du fichier .dsc, tapez la commande :

```
sudo pbuilder build *.dsc
```

Pbuilder va alors compiler le paquet en suivant la description contenue dans le fichier .dsc.

Au final, vous obtiendrez le paquet debian sous la forme d'un fichier en .deb contenu dans `/var/cache/pbuilder/result`.

Pour installer ce nouveau paquet :

```
dpkg -i [nomdupaquet].deb
```

7°) Autres fichiers

a) Fichiers [pre/post][inst/rm]

Les fichiers [pre/post][inst/rm] sont des scripts qui peuvent être exécutés avant ou après l'installation ou la suppression du paquet. Ils permettent donc de déclarer et de modifier des variables d'environnements ou encore de créer des liens symboliques.

Voici un exemple de fichier postinst :



```
1 #!/bin/sh
2
3 echo "export TOM_HOME=/usr/share/tom" >> ~/.bashrc
4 echo "export PATH=${TOM_HOME}/bin:${PATH}" >> ~/.bashrc
5 echo "export CLASSPATH=${TOM_HOME}/lib/tom-runtime-full.jar:${CLASSPATH}" >>
~/.bashrc|
```

Il ne faut pas oublier de faire un fichier postrm afin de supprimer les éventuels liens symboliques créés lors de l'installation.

Enfin, pour rendre le fichier exécutable, il suffit d'exécuter cette commande :

`chmod 755 post*`

b) Fichier dirs

Ce fichier permet de créer des répertoires qui ne sont pas créés automatiquement à l'installation du programme. Il n'est pas nécessaire de créer des dossiers pour le fichier install, ceux-ci seront créés automatiquement.

c) Fichier doc

Ce fichier indique les répertoires contenant la documentation du logiciel empaqueté.