

# Créer des paquetages tar.gz et rpm «à la GNU»

Christophe Pallier\*

12 février 2001

N'avez vous jamais eu envie de distribuer vos projets sous la forme d'un paquetage tar.gz comme ceux du projet GNU<sup>1</sup>? Le premier avantage est que l'installation est normalisée: en général, une simple commande `./configure && make && make install` suffit pour compiler et installer un paquetage (et un `make uninstall` pour tout désinstaller). Le deuxième avantage est que le script `configure` détecte les particularités du système sur lequel s'effectue l'installation. Cela permet d'écrire des programmes qui s'adaptent et compilent sur des systèmes très variés. Même si vous ne programmez pas en C et distribuez de simples scripts, en shell ou awk par exemple, la création d'un paquetage «à la GNU» permet une installation automatique et propre: `configure` détecte où se trouvent les programmes `shell` ou `awk` (dans `/bin`, `/usr/bin` ou `/usr/local/bin...`), et modifie la première ligne des scripts en conséquence. Il peut également installer les documentations (pages `'man'` ou `'info'`) aux endroits appropriés.

Les paquetages qui suivent le standard GNU obéissent à des règles très précises et assez complexes. Il existe bien des outils qui aident à générer de telles distributions: les principaux sont `autoconf` et `automake`. Toutefois, leur documentation est d'un abord quelque peu difficile. Cet article montre, sur un exemple simple, comment les utiliser pour créer des paquetages tar.gz.

Nous expliquerons également, de façon succincte, comment créer un paquetage rpm à partir d'un source tar.gz. De nombreux utilisateurs préfèrent les rpms qu'ils trouvent plus simples à installer (`rpm -i`) et à désinstaller (`rpm -e`). Les principaux intérêts des rpms sont la vérification (`rpm -Va`) qui permet de détecter les fichiers qui ont été modifiés depuis l'installation, et la possibilité, pour un fichier donné, de

---

\*Copyright (c) 2000 Christophe.Pallier@m4x.org Permission est donnée de copier, distribuer et modifier ce document selon les termes de la licence GNU pour les documentations libres, version 1.1, publiée par la Free Software Fondation (cf. <http://www.fsf.org/copyleft/fdl.html>). L'original de ce texte est accessible sur <http://www.pallier.org>.

1. La page d'accueil du projet, en français: <http://www.gnu.org/home.fr.html>

déterminer de quel paquetage rpm il provient (*rpm -qif*). L'envers de la médaille est qu'il faut nécessairement être super-utilisateur pour les installer. Plus gênant encore, les fichiers rpms contenant des exécutables dépendent très fortement de la distribution pour laquelle ils ont été créés (attention donc, si vous installez sur votre système un paquetage rpm destiné à un autre...).

Notre premier projet est un simple programme en C, consistant en un unique fichier source nommé, comme par hasard, *hello.c*. En premier lieu, il faut créer un répertoire pour le projet (*mkdir hello*). A l'intérieur de ce répertoire, nous créons un sous-répertoire (*mkdir src*) dans lequel sera placé le fichier *hello.c*, listé dans la table 1.

TAB. 1 – Fichier source 'hello.c'

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <stdio.h>

int
main()
{
    printf("%s %s\n", PACKAGE, VERSION);
    printf("Hello world!\n");
    printf("Size of int = %d bytes\n", sizeof(int));
#ifdef WORDS_BIGENDIAN
    printf("Big endian\n");
#else
    printf("Small endian\n");
#endif
}
```

Cette version de *hello.c* contient quelques lignes de plus que le programme classique. Ces lignes supplémentaires ont pour but d'illustrer quelques possibilités d'autoconfiguration. Le fichier inclus *config.h* sera généré par l'appel *configure*, il contiendra, entre autres, la définition de macros qui spécifient des caractéristiques de la machine où s'effectue l'installation; par exemple, *WORDS\_BIGENDIAN* spécifie l'ordre des octets dans les mots, et *sizeof(int)* spécifie la taille, en octets, occupé par les variables de type 'int'.

Pour générer le projet, il faut maintenant créer quatre fichiers: *Makefile.am*, *src/Makefile.am*, *acconfig.h* et *configure.in* (cf. Table 2). *Makefile.am* indique les

sous-répertoires à traiter. *src/Makefile.am* indique les binaires à produire et leurs sources. *acconfig.h* est l'embryon qui servira à produire le fichier *config.h*. Finalement, le fichier le plus substantiel est *configure.in*.

TAB. 2 – *Fichiers de configuration*

Fichier	Contenu
Makefile.am	<pre>SUBDIRS = src</pre>
src/Makefile.am	<pre>bin_PROGRAMS = hello hello_SOURCES = hello.c</pre>
acconfig.h	<pre>#undef PACKAGE #undef VERSION</pre>
configure.in	<pre>AC_INIT(src/hello.c) AM_INIT_AUTOMAKE(hello, 0.1) AC_PROG_CC AM_CONFIG_HEADER(config.h) AC_C_BIGENDIAN AC_CHECK_SIZEOF(int) AC_OUTPUT(Makefile src/Makefile)</pre>

Une fois ces fichiers créés, il faut générer différents fichiers de support, en exécutant les commandes suivantes (les caractères '#' introduisent des remarques; seules les commandes, situées à leur gauche, doivent être tapées).

```
aclocal      # create aclocal.m4
autoheader  # create config.h.in
automake -a  # create Makefile.in & src/Makefile.in
autoconf    # create configure
```

Ne vous inquiétez pas d'éventuels messages d'avertissement. Par contre si une commande est inconnue, vérifiez que votre système contient bien les outils de base de développement GNU tels que le compilateur gcc, make, autoconf, automake.

Nous pouvons maintenant simuler une compilation et une installation:

```
./configure # create Makefile
make        # compile the package
make install # install the package
```

Il faut noter que, par défaut, les fichiers sont installés dans */usr/local*, et donc que l'étape '*make install*' nécessite généralement d'être superutilisateur. Néanmoins, on peut installer les programmes à un autre endroit: en exécutant la commande *configure*, avec l'option *--prefix=/home/user*, les fichiers s'installeront dans des sous-répertoires *bin*, *man...*,etc. du répertoire de l'utilisateur '*user*'. (le fichier *INSTALL* liste d'autres autres options d'installation). C'est l'un des avantages importants de ces paquetages *tar.gz*, par rapport aux *rpms*, que de pouvoir être installés par un utilisateur qui n'a pas les droits de '*root*'.

Pour créer un paquetage respectant à la lettre les standards GNU, il est nécessaire de créer des fichiers *NEWS README AUTHORS* et *ChangeLog* dans le répertoire principal du projet. Les informations qui doivent apparaître dans ces fichiers sont précisées dans la documentation info *GNU coding standards*, mais les gens pressés pourront se conformer à cette exigence par un simple '*touch NEWS README AUTHORS ChangeLog*'. Pour générer le package *hello-0.1.tar.gz*, il ne reste alors plus qu'à taper:

```
make distclean # clean the projet
make dist      # create hello-0.1.tar.gz
```

Essayez ensuite de décompacter le paquetage, de le compiler et de l'installer. Si vous avez bien suivi la recette, tout devrait fonctionner «automatiquement».

Quelques remarques méritent d'être mentionnées. Le fichier le plus complexe est *configure.in*; cependant il existe une commande, *autoscan*, qui accomplit une partie du travail en produisant un fichier *configure.scan* pouvant servir de point de départ pour l'écriture de ce fichier. Le numéro de version (ici 0.1) provient de la ligne *AC\_INIT\_AUTOMAKE* de *configure.in*. Il est accessible dans le source par la macro '*VERSION*'. Une fois que vous avez exécuté *./configure* et que le fichier *Makefile* existe dans votre répertoire, il n'est plus nécessaire d'entrer toute la série de commandes *aclocal...* à chaque fois que vous aurez modifié un fichier (par exemple *configure.in*): une simple commande '*make*' suffira.

TAB. 3 – Fichier *src/Makefile.am* installant des scripts *gawk*

```
bin_SCRIPTS = script1 script2
CLEANFILES = $(bin_SCRIPTS)
EXTRA_DIST = script1.awk script2.awk

script1 : script1.awk
        echo "#! " $(GAWK) > $@
        cat $< >> $@
        chmod ugo+x $@

script2 : script2.awk
        echo "#! " $(GAWK) > $@
        cat $< >> $@
        chmod ugo+x $@
```

## Installer de la documentation ou des scripts

Pour installer des pages de manuel, il faut ajouter dans *Makefile.am* une ligne commençant par `man_MANS =` et suivie de la liste des fichiers au format nroff (p.ex. `man_MANS = hello.1`). Nous conseillons de mettre tous ces fichiers dans un sous-répertoire *man* de la distribution, et d'ajouter ce répertoire dans la liste `SUBDIRS` du fichier *Makefile.am* principal.

Supposons maintenant que notre distribution contiennent des scripts écrit dans le langage *gawk*: *script1.awk* et *script2.awk*. Nous voudrions que le chemin de *gawk* soit détecté automatiquement et que ces scripts soient installés, sous les noms *script1* et *script2*. Pour cela, il faut modifier *configure.in* et *src/Makefile.am*. Ajoutez une ligne «`AC_PATH_PROGS(GAWK, gawk)`» dans *configure.in*; cela vérifiera la disponibilité de *gawk* sur le système, et fournira son chemin dans la variable `$(GAWK)`. Le nouveau *src/Makefile.am* est listé dans la table 3. Cet exemple peut facilement être adapté à des scripts écrits dans des langages quelconques, *tcl*, *perl* ou *python*, par exemple.

## Créer un paquetage rpm

Certains utilisateurs préfèrent les paquetages rpm aux tar.gz. Il y en fait deux types de rpm, ceux qui contiennent les sources (*src.rpm*) et ceux qui ne contiennent que les exécutables. Les seconds sont effectivement faciles à installer puisqu'ils ne

nécessitent pas de compilation; mais ils font des hypothèses sur la machine sur laquelle ils s'installent, et sont donc généralement spécifiques à une distribution.

Créer des paquetages src.rpm et rpm à partir de *hello-0.1.tar.gz* n'est pas difficile. La procédure générique est détaillée dans des documents disponibles sur les sites [www.rpm.org](http://www.rpm.org) et [www.rpmdp.org](http://www.rpmdp.org). Essentiellement, il s'agit de créer un fichier texte, ayant pour extension *.spec*, qui décrit le paquetage. Pour 'hello', voici la marche à suivre:

1. copier *hello-4.1.tar.gz* dans `/usr/src/packages/SOURCES` (ou `/usr/src/redhat/SOURCES` selon votre distribution)
2. dans `/usr/src/packages/SPECS`, créer le fichier *hello.spec* (cf. table 4). Ce fichier décrit le contenu du futur paquetage rpm.
3. faire 'rpm -ba hello.spec' (en tant que root)

La commande 'rpm -ba' crée le fichier *hello-0.1-1.rpm* dans `RPMS/i386` et *hello-0.1-1.src.rpm* dans `SRPMS`. Si vous distribuez l'exécutable, un superutilisateur pourra effectuer les opérations suivantes:

```
rpm -qip hello-0.1-1.rpm # liste le contenu du paquetage
rpm -i hello-0.1-1.rpm # installe
rpm -qil hello # vérifie le paquetage
rpm -e hello # désinstalle
```

## Conclusion

Vous voici maintenant parti du bon pied pour créer vos paquetages. Pour faire des choses plus complexes, vous devrez lire les documentations au format info: GNU coding standards, Autoconf, Automake. Le lecture des fichiers *configure.in* des paquetages existants est également instructive.

TAB. 4 – *Listing du fichier 'hello.spec' pour la construction des rpms*

```
#
# exemple de fichier 'spec' pour le projet 'hello'
#
Vendor:      Christophe Pallier
Name:        hello
Release:     1
Copyright:   GPL (cf http://www.fsf.org)
Group:       unsorted
Provides:    hello
Packager:    chrplr@pallier.org

Version:     0.1
Summary:     The hello program
Source:      hello-0.1.tar.gz

%description
A slightly adapted version of the 'hello' program.

%prep
%setup
%build
./configure
make
%install
make install
%files
/usr/local/bin/hello
```