Activité : déploiement d'une application (script autoextractible)

Thierry Vaira <tvaira@free.fr>

Table des matières

Déploiement d'une application (script autoextractible)

Expression du besoin

Dans le cadre du **déploiement** d'une application, on a besoin de réaliser un **script d'installation** de celle-ci.

Déploiement

Pour réaliser une procédure d'installation d'une application, il faudra décomposer celle-ci en trois parties :

- fabriquer une version finale (*release*) de l'application (`make`) et un script d'installation (`setup.sh`)
- fabriquer un installateur auto-extractible pour l'application (`makeself`)
- déployer cette application sur une machine à partir d'un script auto-extractible (`setup`)

Ces trois parties sont parfois dépendantes de la plateforme et des outils utilisés.

Une autre solution consisterait à réaliser un **paquet** ou paquetage (*package*) pour la distribution de votre choix.

```
Dans le contexte des systèmes UNIX, on appelle paquet (ou parfois paquetage, en anglais package) une archive (
    fichier compressé) comprenant les fichiers informatiques, les informations et procédures nécessaires à l'
    installation d'un logiciel sur un système d'exploitation au sein d'un agrégat logiciel, en s'assurant de la
     cohérence fonctionnelle du système ainsi modifié. Actuellement, on distingue deux types de paquets : deb
    qui est le format des paquets logiciels de la distribution Debian GNU/Linux (et presque toutes les
    distributions basées sur Debian) et rpm (Red Hat Package Manager) qui est un format ouvert initié par la
    distribution Red Hat GNU/Linux.
```

Voir aussi : Tutoriel RPM

Fabriquer une version finale de l'application (release)

De manière générale, il faut résoudre un certain nombre de problèmes et répondre à quelques choix.

- Droits et licence ? Les droits et les fichiers à déployer (consulter le contrat de licence de Qt par exemple) et les droits et les fichiers à appliquer à son application (indispensable de lire www.gnu.org/licenses/).
- Statique ou dynamique ? Il faudra choisir entre fabriquer un exécutable indépendant (statique) ou non (dynamique). Sous Linux, les dépendances sont nombreuses : architecture multi-plateforme, bibliothèques, etc ...
- Librairies dynamiques ? Les bibliothèques dynamiques contiennent du code exécutable (des fonctions formant une API) qui sera susceptible d'être utilisé par un (ou plusieurs) programmes au moment de leur exécution. En cas de besoin, la librairie dynamique sera chargée en mémoire et son code sera alors utilisable par le programme demandeur. Il en résulte les avantages suivants : la taille du programme est réduite (puisque le code dont il a besoin se trouve dans la librairie), la possibilité de faire évoluer la librairie sans avoir à recompiler (si le prototype des fonctions définies dans la librairie reste inchangé). L'inconvénient majeur reste l'obligation de la présence de la librairie sur le système cible pour que le

programme puisse s'exécuter. L'extension usuelle d'une librairie dynamique sous Linux est `.so` (pour Windows, ce sera .dll)
- Dépendances ? Il y a plusieurs techniques pour connaître les dépendances externes d'une application : utiliser un utilitaire qui recherche ces dépendances (la commande `ldd` sous Linux) ou appliquer une démarche (rudimentaire !) qui teste l'application sur une machine vierge (une machine virtuelle par exemple) et qui résout les dépendances les unes après les autres.
- L'emplacement ? Il faudra déterminer où installer l'application (dans les chemins partagés `/usr/bin/` et `/usr/local/bin` ou séparés dans `/opt/` pour Linux). On peut aussi envisager de créer une entrée dans le menu Application de l'environnement de bureau et aussi un raccoruci.
- Les fichiers de l'application à déployer ? Il faut évidemment recenser les fichiers (ainsi que l'arborescence) qui composent l'application et qui devront être déployer avec celle-ci. De manière générale, on trouve : l'exécutable, l'icône (`.ico`), des fichiers de configuration (comme les `.ini`), des fichier multimédia (comme des images), des fichiers d'aide (comme les `.chm` ou les pages `man`, des fichiers `.html`, ...), un fichier licence, un fichier readme, etc ...

On désire au final regrouper toutes ces actions et tous ces fichiers dans un même et seul fichier exécutable : l'installateur.

Fabriquer un installateur auto-extractible pour l'application

Pour le déploiement de l'application, notre choix se porte sur l'utilitaire makeself.

Makeself est un petit script *shell* qui génère une archive auto-extractible (`tar`) d'un répertoire. Le résultat apparait comme un script *shell* exécutable. L'archive va ensuite s'auto-décompresser et lancer un installateur de votre choix (un autre script par exemple). Cela est assez similaire aux archive Winzip auto extractibles. Il gère aussi l'intégrité via un checksum CRC et/ou MD5 (les deux par défaut).

A ce jour, ce script est utilisé par les entreprises suivantes :

- Id Software (Editeur de jeux vidéo : Quake 3, Return To Castle Wolfenstien) .
- Loki Software pour les jeux qu'il édite.
- Les drivers pour Linux du constructeur nVidia.
- L'installateur de Google Earth pour Linux
- Le package Makeself lui-même.

```
Un fichier auto-extractible, également connu sous le sigle SFX (self-extracting archive), est un type de fichier
     informatique compressé qui contient en lui-même les outils nécessaires à sa propre décompression, de sorte
     que contrairement aux autres fichiers compressés, il n'est pas nécessaire de disposer d'un logiciel de
    compression pour accéder aux données qu'il contient. Il s'agit d'un fichier exécutable qui contient la
    charge utile. Il suffit généralement d'en changer l'extension (*.exe sous Windows ou .bin sous Linux) et de
     double-cliquer dessus pour que la décompression s'opère. Naturellement il faut que ce fichier exécutable
    soit exécutable sous le système d'exploitation considéré.
```

Vous devez avoir installé au préalable l'archive `makeself.run`. Vous pouvez installer, par exemple, makeself dans le répertoire `/usr/local/`.

```
# ./makeself.run
```

Exemple

On désire créer un fichier auto-extractible `setup.sh` (ou `install.sh`) contenant :

- les fichiers de l'application :

```
./tp1/AUTHORS
./tp1/README
./tp1/NEWS
./tp1/ChangeLog
./tp1/INSTALL
./tp1/COPYING
./tp1/TODO
./tp1/usr/share/man/man1/qhelloworld.1.lzma
./tp1/usr/bin/qhelloworld
```

- le script d'installation à exécuter au lancement :


```
./tp1/setup.sh
```

Vous pouvez récupérer ces fichiers dans l'archive et la désarchiver avec la commande :

```
$ tar zxvf qhelloworld-1.0.0-makeself.tar.gz
```

Archive du tp : qhelloworld-1.0.0-makeself.tar.gz

Pour pouvoir installer correctement l'application, il suffit d'éditer le script d'installation `setup.sh` et de paramétrer les variables à sa convenance.

```
$ vim setup.sh
### Informations sur l'installation
NAME="qhelloworld"
VERSION="1.0.0"
SUMMARY="Affiche \"Hello world !\" dans une boite de dialogue"
LICENCE="GPL"
AUTEUR="Thierry Vaira <thierry.vaira@orange.fr>"

HAVE_MENU_KDE="yes"
ICON="qdevelop"
CATEGORY="Qt;Development;IDE;"
MIMETYPE="application/x-qdevelop;"
####################################

### liste des ressources à installer
BIN_FILES="${NAME}"
DOC_FILES="README NEWS COPYING AUTHORS TODO ChangeLog"
MAN_FILES="man1/*"
MAN_FILES="man1/qhelloworld.1.*"
EXTRA_FILES=""
####################################

### repertoires
BINDIR="usr/bin"
DATADIR="usr/share"
DOCDIR="${DATADIR}/doc/${NAME}"
MANDIR="${DATADIR}/man"
EXTRADIR="${DATADIR}/${NAME}"
TMPDIR="/tmp"
```

On peut tester manuellement avant de construire l'installateur :

```
# ./tp1/setup.sh
```

Pour construire l'installateur, il suffit de taper une seule commande :

```
# ./makeself.sh ./tp1 setup.sh "Script d'installation by tv (bts sn-ir)" ./setup.sh
```

Les arguments passés ici à makeself sont :

- `./tp1` : le répertoire qui contient les fichiers à « empaqueter » (qui seront ajoutés à l'archive)
- `setup.sh` : le nom de l'installateur auto-extractible (le nom de l'archive)
- "blabla" : un label d'identification
- `./setup.sh` : le nom du script qui sera automatiquement exécuté au lancement

On peut aussi consulter l'aide en ligne de `makeself` :

```
Usage: ./makeself.sh [params] archive_dir file_name label [startup_script] [args]
...
```

Lire : www.megastep.org/makeself/

Avant d'installer l'application, on va récupérer des informations sur l'installateur.

On exécute le script avec les options suivantes :


```
# ./setup.sh --info
Identification: Script d'installation by tv (bts sn-ir)
Target directory: tp1
Uncompressed size: 484 KB
Compression: gzip
Date of packaging: Sat Apr 17 14:27:04 CEST 2010
Built with Makeself version 2.1.5 on linux-gnu
Build command was: ./makeself.sh \
    "./tp1" \
    "setup.sh" \
    "Script d'installation by tv (bts sn-ir)" \
    "./setup.sh"
Script run after extraction:
     ./setup.sh
tp1 will be removed after extraction

# ./setup.sh --list
Target directory: tp1
drwxr-xr-x root/root         0 2010-04-17 14:26 ./
-rwxr-xr-x root/root     14849 2010-04-17 14:24 ./setup.sh
-rw-r--r-- root/root        54 2010-04-17 14:24 ./AUTHORS
-rw-r--r-- root/root        20 2010-04-17 14:24 ./README
-rw-r--r-- root/root         0 2010-04-17 14:24 ./NEWS
-rw-r--r-- root/root       266 2010-04-17 14:24 ./ChangeLog
-rw-r--r-- root/root      9240 2010-04-17 14:24 ./INSTALL
-rw-r--r-- root/root     17992 2010-04-17 14:24 ./COPYING
-rw-r--r-- root/root        28 2010-04-17 14:24 ./TODO
drwxr-xr-x root/root         0 2010-04-17 14:24 ./usr/
drwxr-xr-x root/root         0 2010-04-17 14:24 ./usr/share/
drwxr-xr-x root/root         0 2010-04-17 14:24 ./usr/share/man/
drwxr-xr-x root/root         0 2010-04-17 14:24 ./usr/share/man/man1/
-rw-r--r-- root/root       489 2010-04-17 14:24 ./usr/share/man/man1/qhelloworld.1.lzma
drwxr-xr-x root/root         0 2010-04-17 14:24 ./usr/bin/
-rwxr-xr-x root/root    387314 2010-04-17 14:24 ./usr/bin/qhelloworld

# ./setup.sh --check
Verifying archive integrity... MD5 checksums are OK. All good.
```

Pour installer le paquetage (sous *root*) :

```
# ./setup.sh
Verifying archive integrity... All good.
Uncompressing Script d'installation by tv (bts sn-ir) ..........
```

Le script d'installation prend ensuite le relais pour assurer l'installation ...

On peut tester :

```
$ qhelloworld

$ man qhelloworld
```

Puis, pour désinstaller (sous root) :

```
# uninstall-qhelloworld-1.0.0.sh
```

Et bien évidemment, cela ne marche plus :

```
$ qhelloworld
bash: /usr/bin/qhelloworld: Aucun fichier ou dossier de ce type

$ man qhelloworld
Il n'y a pas de page de manuel pour qhelloworld.
```

Retour au sommaire

