# Département Informatique

**R5.A.07 Automatisation de la chaîne de production (distanciel)**

**TP Package 1**

Responsable : X.Roirand

Durée : 180mn machine

---

Dans le cadre de la CI/CD (cherchez sur internet si vous ne savez pas ce que cela veut dire), on est souvent amené à packager des logiciels produit, pour en faire des produits ou des choses installables.

Lors de cette CI/CD, il va y avoir plusieurs étapes, mais le but du jeu est de produire quelque chose qui soit le plus fidèle possible aux sources des programmes et fonctionnel. Pour appréhender cet objectif, je vous propose de produire quelque chose d'équivalent, en utilisant la notion de package, qui permet, sous Unix, de rassembler dans un ensemble self contained (ou avec des dépendances)

Le but de ce TP va être de créer un package sur une machine, et de tester que l'installation du package fonctionne et de comprendre à minima ce qu'on peut faire lors de l'installation d 'un package.

## 1) Connexion sur la machine distante sur laquelle on doit travailler

Pour se connecter sur la machine distante, il faut utiliser une connexion ssh. C'est une connexion sécurisée qui permet de ne pas faire passer le mot de passe en clair sur le réseau.

Vous pouvez utiliser le programme « putty » sur windows ou « ssh » sur linux.

La connexion se fait en 2 étapes, dans le cadre de nos TP. il faut d'abord se connecter sur une sandbox dont l'adresse IP est 149.202.85.73. Le login est student et le mot de passe ?Student_56. Une fois sur cette sandbox, connectez-vous sur votre machine distante, pour cela il faut connaitre le numéro que l'enseignant vous a donné, et ajouter ce numéro au nom : machine pour obtenir le nom de la machine sur laquelle vous devez vous connecter.

Exemple : si votre numero est le 13 alors il faut faire un ssh sur la machine machine13

Ensuite le login et mot de passe sur votre machine distante sont :

login : root
mot de passe : ?Student_56

donc si votre numero est le 13 alors le mot de passe est ?13Student 56, comme suit :

login : root
mot de passe : ?13Student_56

Une fois sur votre machine distante, vous pouvez l'administrer totalement.

Pour pouvoir noter le TP, il faut que vous mettiez dans un fichier nommé /root/info.txt votre nom et prénom.

## Attention, pas de fichier = zéro !

Créez un user test avec le mot de passe que vous voulez. Pensez à vérifier qu'il ait bien une HOME directory.

Toute la suite du TP doit être fait en tant que student.

## 2) Qu'est ce qu'un RPM ?

Les .rpm sont des fichiers permettant d'installer des packages. Par package, on entend n'importe quel ensemble de fichiers (exécutables ou non) qui peuvent se greffer au système. Il peut s'agir de fichiers de configuration, d'une documentation, du noyau d'un système, ou de logiciels.
Le logiciel rpm est un gestionnaire de packages qui permet de consulter, supprimer, rechercher, etc. des packages. Sur ubuntu, qui est le systeme de votre machine, on utilise dpkg qui fait à peu près la même chose que rpm.
Lors de l'installation, il prend en argument un fichier .rpm et on peut donc effectuer l'installation comme suit :

```
[klaus@localhost rpmbuild]$ su
Password:
[root@localhost bin]# dpkg –iv domino-1-1.noarch.rpm
Préparation... ########################################### [100%]
1:domino ########################################### [100%]
```

Vous remarquerez qu'il est nécessaire d'être root pour installer un package, les options -iv servent à préciser qu'il s'agit d'une installation (-i pour install, et -v pour verbose), et domino-1-1.noarch.rpm est le nom du fichier .rpm contenant le package.

## 3) Comment construire un RPM ?

On va se servir de rpmbuild pour construire un package. rpmbuild est un logiciel permettant de générer des rpms. Regarder un peu sur internet quelques minutes comment fonctionne rpmbuild (Attention, si vous ne faite pas cette partie là, vous allez galérez ensuite pour construire votre package).

## 4) Construction du package (programme source)

Nous allons ensemble créer un .rpm pour le programme helloWorld, qui sera écrit en C. Je suppose que vous êtes un minimum familier avec la programmation mais que vous n'avez jamais créé de rpms.

Passez en tant qu'utilisateur test. Tout le reste du TP se fera en tant que test, sauf dans les moments ou vous devrez installer un package sur votre machine ou là vous serez obligé d'être super utilisateur.

Trouvez un programme helloworld sur internet écris en C.

Compilez le (ca ne fonctionnera peut-être pas directement, vous aurez sûrement des choses à installer pour celà fonctionne).

Executez le, il doit afficher le message helloworld à l'écran.

## 5) Construction du package (répertoires)

Nous devons faire la la distinction entre deux types de packages .rpm :

– les binaires contiennent les fichiers déjà compilés, ils dépendent de la plate-forme sur laquelle vous souhaitez installer votre package. Ils portent l'extension .XXX.rpm ou XXX est la plate-forme (i586, noarch, etc).

– less sources contiennent les sources et les instructions nécessaires pour que la compilation puisse se faire automatiquement sur la plateforme de l'utilisateur. On a cette fois-ci l'extension .src.rpm.

Pour construire le RPM du TP, une fois qu'on a le programme source (helloworld), il faut installer rpmbuild, qui va nous server a construire le package et il faut créer l'arborescence nécessaire pour construire le package.

L'installation de rpmbuild va se faire en installant le package qui contient rpmbuild, a vous de jouer 😊.

**Principe de rpmbuild:**

Nous allons dans la partie suivante générer un package binaire. Dans ce cas rpmbuild exécute toutes les instructions nécessaires à la compilation et à l'installation de votre programme. Une fois que votre programme est compilé et installé, rpmbuild prend une "photographie" de vos répertoires. Lors de l'installation, cette photographie est recopiée dans l'arborescence du système de l'utilisateur. Par photographie, on entend une liste de fichiers ainsi que les réepertoires dans lesquels ils devront être recopiés.

**Voici un pseudo-exemple pour vous aider à comprendre:**

> Vous avez créé un fichier de description rpmbuild qui va indiquer à rpmbuild comment construire le rpm voulu. Ensuite vous avez lancé rpmbuild pour construire ce rpm et vous obtenez (à l'endroit que vous avez indiqué dans le fichier de description rpmbuild) l'arborescence suivante:
>
> /home/test/mybuild:
> i686/usr/doc/README.txt
> i686/usr/bin/myclient
> i686/usr/bin/myserver
>
> Le pseudo root indiqué ici est /home/test/mybuild/i686, ce qui veut dire que lorsque le fichier rpm final est buildé, rpmbuild va allez prendre tout ce qui se trouve sous / home/test/mybuild/i686 et en fait un fichier comme un fichier archive, mais à partir de cet endroit, ce qui va doner comme contenu:
>
> /usr/doc/README.txt
> /usr/bin/myclient

/usr/bin/myserver

Et ca sera cette arborescence là qui va ensuite etre dans le fichier rpm et qui va être installée sur le système final, celui sur lequel on va installer le fichier RPM lorsqu'on fera un rpm -i ….<lefichier.rpm>

Donc lorsque vous allez construire le fichier rpm, vous pourrez inspecter son contenu avec la commande "rpm -qlp <votre_fichier_rpm>"

**Création des répertoires**

La première étape est de faire la mise en place, c'est à dire de créer un répertoire qui va contenir l'ensemble de votre travail, pour celà vous allez créer un répertoire $HOME/ rpmbuild (vous pouvez l'appeler $HOME/RPM, $HOME/toto ou $HOME/monrepertoire si ,ca vous chante) qui servira de "laboratoire" pour la compilation et l'installation. Pour que rpmbuild, fonctionne correctement, il convient de lui indiquer quelque part le chemin du laboratoire. On peut le faire soit dans l'environnement avant de faire le build, soit une fois pour toute dans votre environement en le mettant dans un fichier de configuration, et c'est cette option que vous allez prendre.

Editez le fichier $HOME/.rpmmacros et placez-y les données suivantes :

%_topdir /home/test/rpmbuild
%_tmppath /home/test/rpmbuild/tmp

On suppose ici quel la home de test est /home/test sinon il faut adapter avec vos valeurs.

Le répertoire $HOME/rpmbuild/tmp sera utilisé par rpmbuild pour placer des fichiers qui seront détruits après la création du package. Vous devez ensuite créer un ensemble de répertoires :

mkdir -p ~/rpmbuild/SOURCES ~/rpmbuild/SPECS ~/rpmbuild/RPMS ~/rpmbuild/SRPMS ~/rpmbuild/tmp ~/rpmbuild/BUILDROOT

Passons en revue ces répertoires:

– $HOME/rpmbuild/SOURCES : contient les fichiers sources de l'application.
– $HOME/rpmbuild/RPMS : contiendra le rpm binaire à la fin de l'exécution de rpmbuild.
– $HOME/rpmbuild/SRPMS : contiendra le rpm source à la fin de l'exécution de rpmbuild.
– $HOME/rpmbuild/SPECS : contient un fichier de configuration qui sera utilisé par rpmbuild pour determiner comment compiler les sources, les installer, quelles sont les dépendances, le nom de l'application, etc.
– $HOME/rpmbuild/BUILDROOT : répertoire qui servira de laboratoire, les binaires y seront placés pour la photo.

**Fichier de specification**

La phase délicate est l'élaboration du fichier de spécification, appelé généralement spec. Ce fichier est en deux parties, que j'appellerai l'entête et les commandes.

**Entête**

Le fichier de spécification contient les informations suivantes dans son entête :

– BuildRoot : chemin vers le laboratoire
– Summary : description du package en une ligne
– Licence : type de licence (GNU, GPL, ...)
– Name : nom du package
– Version : version du package
– Release : release du package
– Group : catégorie du package (Education, Developpment, ...)
– BuildArchitecture : architecture de la machine de l'utilisateur (i586, noarch, ...)
– %description : description détaillée du package

Ces informations, techniquement, n'indiquent pas à rpmbuild comment construire le package. Il s'agit simplement de données qui aideront les utilisateurs à s'y retrouver. A noter que ces informations seront quand même stockées sous une forme ou une autre dans le rpm final.

**Commandes**

Ensuite, on crée des sections contenant les commandes qui vont être exécutées aux différentes étapes de la construction du rpm.

– %build : commandes de compilation des sources
– %install : commandes d'installation des binaires
– %clean : commandes pour faire le ménage sur votre machine, une fois le package construit
– %files : liste des fichiers (avec le chemin complet) tels qu'ils seront recopiés sur la machine de l'utilisateur.

Allez chercher sur internet comment se déroule les différentes étapes afin de gagner en compréhension dans la suite.

**Création et installation de hello-world**

**Construction du package**

Voici le fichier de spécification conçu pour le package hello-world que je vous demande de prendre et de copier sur votre machine distante.

```
# This is a sample spec file for helloWorld
BuildRoot: %{_topdir}/BUILDROOT/
Summary: Prints "Hello world!" on stdout
License: GNU
Name: hello-world
Version: 1
```

*Release: 1*
*Group: Education*
*BuildArchitectures: i586*
*%description*
*Prints "Hello world!" on stdout, that's all !*
*%build*
*4cd %{_sourcedir};*
*gcc -Wall -o helloWorld helloWorld.c*
*%install*
*rm -fR $RPM_BUILD_ROOT;*
*mkkdir -p $RPM_BUILD_ROOT/usr/bin;*
*cd %{_sourcedir}*
*mv helloWorld $RPM_BUILD_ROOT/usr/bin/;*
*%clean*
*rm -rf $RPM_BUILD_ROOT/*
*%files*
*%attr(755,root,root)*
*/usr/bin/helloWorld*

Je vous le donne c'est cadeau 😊 bon…il se peut que j'ai fais une ou deux boulettes dedans… vous verrez bien à l'usage.

Avant de le passer en revue nous allons l'exécuter. Pour ce faire, il convient de placer correctement dans l'arborescence tous les fichiers intervenant dans la création du package.
Copiez le fichier helloWorld.c dans ~/rpmbuild/SOURCES/
Copiez le fichier spec dans ~/rpmbuild/SPECS/

On génère ensuite le package contenant les binaires en exécutant la commande rpmbuild -v -bb $HOME/rpmbuild/SPECS/spec.
L'option -v sert `a rendre rpmbuild "verbeux", -bb précise que l'on souhaite créer un package binaire, $HOME/rpmbuild/SPECS/spec est le chemin du fichier de spécification.

Construisez le package avec la commande rpmbuild

Vérifiez que tout est bon dans l'output généré sur l'écran. Si ca n'est pas bon, cherchez le ou les erreurs (une à la fois c'est déjà pas mal), corrigez les jusqu'à ce que tout se déroule sans problème.

Une fois cela fait vous n'avez plus qu'à installer le rpm que vous trouverez dans $HOME/ rpmbuild/RPMS/i586/hello-world-1-1.i586.rpm.

Pour l'installer sur votre systeme debian/ubuntu (machine distante), il vous faut un package de type .deb et non pas .rpm donc il va falloir traduire le fichier rpm en deb, pour cela il faut utiliser l'utilitaire "alien".

Installez le sur votre machine distante et transformez le fichier rpm en fichier deb

Installez le fichier deb généré.

Vérifier que l'executable helloWorld a bien été installé sur le système et que vous pouvez l'exécuter de n'importe ou sur le système.

On va maintenant modifier le rpm pour qu'il contienne, non pas juste un binaire mais aussi un fichier source. Dans la vraie vie on ne fait jamais un mix de binaire et de source dans un rpm mais là on le fait juste pour le TP. Donc modifiez le fichier spec pour que le fichier helloWorld.c soit recopié dans /usr/src.

Générez le rpm et convertissez le en deb et installez le. Vérifiez que maintenant vous avez bien votre binaire helloworld sous /usr/bin et aussi votre fichier source helloWorld.c sous /usr/src.

**De plus en plus fort:**

Vous allez maintenant faire un nouveau fichier SPEC qui va permettre de faire fonctionner un programme helloworld java. Pour cela vous aller devoir faire un nouveau fichier SPEC et donc forcément donner dans la ligne de commande du rpmbuild le nom du fichier SPEC que vous venez de créer. A noter qu'il va vous falloir installer un compilateur java pour compiler le programme et aussi tester qu'il fonctionne. Du coup, au final, une fois le package installé, vous devez avoir le fichier helloworld.class sous /tmp (c'est le répertoire final d'install, c'est bizarre mais c'est juste pour le TP, normalement c'est plus souvent /usr/bin). Donc quand vous tapez java -classpath /tmp helloworld cela devrait fonctionner.

Une fois que tout fonctionne bien vous devez désinstaller le compilateur java et tout environnement java de type JRE, puis vous devez vous débrouiller pour qu'un environnement de type java JRE soit automatique installé quand on installe le package. Une fois que c'est fait, vérifier que vous arrivez à exécuter le programme helloworld java.

L'étape suivante consiste a reprendre le fichier SPEC de l'exemple helloworld en C et a faire que l'exécutable généré ne soit pas en 64 bits mais en 32 bits. Assurez-vous que vous avez bien généré un fichier 32 bits.

La dernière étape consiste en la création de votre propre répository. Jusque là quand vous faite un apt-get install … cela va prendre un package dans un des répository Ubuntu sur internet. Vous allez créer votre propre répository et l'ajouter à la liste des répositories sources de votre machine et copier dedans le premier package helloworld.  En faisant comme ca, vous devriez pouvoir installer le package helloworld en faisant: apt-get install helloworld. Vérifiez que cela fonctionne.