


Chapitre 10 : Logiciels

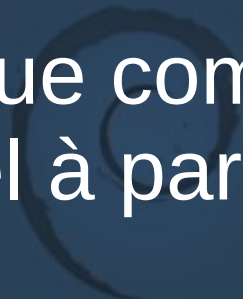
Installation de logiciels dans Linux

The Debian logo, which is a stylized black spiral, is positioned behind the text 'Installation de logiciels dans Linux'.

Debian

10.1 – Installation manuelle

Cette partie explique comment on compile et installe un logiciel à partir de ses sources

The Debian logo, which is a stylized white swirl on a dark blue background, is positioned behind the text.

Debian

Un logiciel Unix ?

- Un logiciel est composé de différents fichiers :
 - Exécutable(s) dans /usr/bin
 - Librairies dans /usr/lib
 - Données : exemples, modèles... dans /usr/share
 - Configuration dans /etc
 - Documentation dans /man, /usr/share/doc
- Les logiciels personnels sont mis dans /usr/local, ou /opt

Fichiers « binaires »

- C'est ainsi qu'on nomme les exécutable : programmes et librairies (bibliothèques)
- Librairie = fichier contenant des fonctions déjà compilées et qu'on peut employer dans un prog :
 - libc contient printf, scanf...
 - libm contient sqrt, sin, log
- Pour le développement, ces librairies sont accompagnées de fichiers .h à inclure
 - Ex : stdio.h, stdlib.h, unistd.h... pour libc

Emplacements

- Les bibliothèques binaires sont situées dans :
 - /lib/x86_64-linux-gnu : bibliothèques de base Unix
 - /usr/lib : bibliothèques rajoutées par les logiciels
 - /usr/local/lib : bibliothèques installées par l'utilisateur
- Ce sont des fichiers nommés `libBidule.so`
 - NB : il y a des liens `libmachin.so.version` vers `libmachin.so`
 - `.so` = shared object : elles sont chargées en mémoire une seule fois et partagées entre les programmes.
 - Emploi : `gcc -o prog source.c -lBidule`
 - NB : la bibliothèque `libc.so` est systématiquement rajoutée

Installer un logiciel

- Ces fichiers sont regroupés dans un fichier qualifié d'**archive** qu'il faut installer :
 - Exemple : un .zip, .tgz ou .tar.bz2
 - C'est un fichier contenant d'autres fichiers
- Procédure :
 - On extrait les fichiers de cette archive
 - On les recopie aux bons endroits
 - Comme cela peut être compliqué, il y a souvent un script d'installation

Logiciel source ou binaire

- En fait, il y a deux sortes de logiciels :
 - Ceux qui sont fournis tout prêts : fichiers **binaires** exécutables, bibliothèques, données...
C'est le cas à chaque fois sur Windows.
 - Ceux (les « open source ») qui sont fournis avec seulement leurs **sources** par exemple en langage C ou C++. Il faut les compiler avant de pouvoir les installer.
On va s'intéresser à ceux-là d'abord.

Point de départ

- On crée un logiciel à partir de fichiers sources
 - On part d'un fichier .c comme ceux faits en algo
`gcc prog.c -o prog`
- Un gros logiciel est composé de nombreux fichiers sources qui sont compilés séparément et regroupés
 - Un seul source serait énorme et ingérable
 - On appelle cela la **compilation séparée**

Exemple de compilation séparée

partie1.c

```
int calc(int v) {  
    int v2 = v * v;  
    return v2 + 1;  
}
```

partie2.c

```
extern int calc(int v);  
int main() {  
    printf("%d\n", calc(3));  
}
```

lien

- (1) gcc -c partie1.c -o partie1.o Compilation
- (2) gcc -c partie2.c -o partie2.o Compilation
- (3) gcc partie1.o partie2.o -o prog Édition des liens

Compilation du logiciel

- Commandes de compilation (le cadre du bas du transparent précédent) :
 - Présence d'options de compilation à ne pas oublier
 - Autres commandes possibles que la compilation
- Possibilités pour faciliter la compilation :
 - Script bash : exactement le cadre du bas précédent
 - Makefile : presque comme le script bash
 - Configure

Makefile

- Un Makefile est un fichier qui *ressemble* à un script, il indique comment compiler :

```
prog:↪ partie1.o partie2.o
↪ gcc partie1.o partie2.o -o prog
partie1.o:↪ partie1.c
↪ gcc -c partie1.c -o partie1.o
partie2.o:↪ partie2.c
↪ gcc -c partie2.c -o partie2.o
```

Compiler avec un Makefile

- Une fois que le Makefile est écrit (avec un éditeur de texte), il suffit de taper la commande **make**
 - Attention le Makefile n'est pas exécutable !
 - **make** lance la construction du logiciel en regardant les commandes inscrites dans le Makefile
 - Vérifie s'il est besoin de compiler certains modules
- Le pb : il faut écrire le fichier Makefile adapté à chaque architecture Unix (il y a des variantes)

Configuration automatique

- Alors, on trouve plutôt un script **configure** :
 - Il vérifie que le système contient toutes les bibliothèques nécessaires pour que le logiciel se compile et fonctionne
 - Il crée automatiquement le fichier **Makefile**
- Mode d'emploi :

```
dsl@box:~/ $ ./configure
```

```
nombreux tests... puis à la fin, génération du Makefile
```

```
dsl@box:~/ $ make
```

Installation automatique

- Le fichier Makefile peut aussi contenir les commandes d'installation et désinstallation :

```
install: prog
    cp prog /usr/bin
uninstall:
    rm -f /usr/bin
prog: partie1.o partie2.o
    gcc partie1.o partie2.o -o prog
... ..
```

Procédure

- Au final, une fois qu'on a les sources du logiciel et que ce logiciel est bien construit (présence du script configure), il suffit de faire ceci :

(1) `./configure`

(2) `make`

(3) `sudo make install`

Le `sudo` final est nécessaire car on modifie le système ; les autres commandes sont faites dans notre compte seulement

Dossiers d'installation

- Quand on procède ainsi, en général, le logiciel se retrouve installé dans /usr/local :
 - Exécutables dans /usr/local/bin
 - Librairies dans /usr/local/lib
 - etc.
- Intérêt : on sait ce qu'on a installé
- Si on veut changer ce chemin :

```
./configure --prefix=/usr
```


Obtention des logiciels

- On s'intéresse aux logiciels « Open Source »
- Les concepteurs les mettent à disposition sous forme d'une archive compressée à télécharger :
 - Avec firefox : *enregistrer la cible du lien sous...*
 - Ou : **wget URL de l'archive**, exemple :
`wget http://lynx.isc.org/current/lynx-cur.tar.gz`
- On récupère un fichier `.tar.gz` (ou similaire)

Extraction des sources

- Une archive tar compressée s'extrait par :
 - Si elle est du type `.tar.gz` ou `.tgz` (compression gzip)
`tar xfvz nom.tgz`
 - Si elle est du type `.tar.bz2` (compression bzip2) :
`tar xfvj nom.tar.bz2`
- On obtient dans tous les cas (sauf si l'archive a été mal faite : pas de dossier englobant) un dossier contenant tous ses fichiers.
 - Il peut y avoir un conflit avec l'utilisateur propriétaire des fichiers.

Bilan

- Au final, voici les étapes :
 - (1) `wget http://serveur/chemin/archive.tgz`
 - (2) `tar xfvz archive.tgz` => dossier *sources*
 - (3) `cd sources`
 - (4) `./configure`
 - (5) `make`
 - (6) `sudo make install`
 - (7) Nettoyage : `cd .. ; rm -fr sources archive.tgz`



Problèmes de cette démarche

- Problèmes potentiels de ce type d'installation :
 - Besoin de certaines librairies ou d'autres logiciels pas forcément installés => installations en cascade
 - Erreurs si organisation des fichiers non standard (choix locaux ou variante Unix)
 - Conflits potentiels avec les logiciels existants
 - Difficulté à désinstaller (cas de fichiers partagés)
 - Difficulté à faire évoluer (versions successives)

10.2 – Gestion de paquets

Cette partie explique la gestion des logiciels par
le système Linux Debian
D'autres systèmes existent (ex : redhat)

Debian

Nécessité d'une gestion globale

- Le mécanisme de paquets **Debian** permet de régler ces difficultés :
 - Procédure d'installation très simple (1 commande)
 - Gestion des dépendances entre logiciels
 - Gestion des versions et des mises à jour
- La totalité du système Linux Debian est gérée par ce mécanisme

Dépôts Debian

- Il y a d'abord une base de données globale dans des serveurs qui sont appelés **dépôts** (*repository*)
 - Exemple : <http://ftp.fr.debian.org/debian/dists/jessie/>
 - Il y en a plusieurs copies (*miroirs*) en cas de pb
- Un dépôt contient le catalogue de tous les logiciels compatibles de la **distribution**
- Il y a plusieurs dépôts : logiciels de base, logiciels non libres, multimédia...

Logiciels Debian

- Un logiciel Debian est appelé **paquet** (*package*)
- L'installation est fiable et sûre :
 - Pas de corruption des données
 - Code de contrôle des fichiers (md5 ou sha)
 - Aucun conflit avec les autres paquets
 - C'est assuré par les administrateurs de la distribution
 - Toutes ses dépendances sont définies
 - Si un paquet a besoin d'une librairie, elle est d'abord installée, et ainsi de suite pour cette librairie

Principe général

- Votre machine se connecte aux dépôts pour obtenir la liste des paquets existants
 - Le dépôt répond « il y a tels paquets avec telles versions, ils font telles tailles et ils dépendent de ces autres paquets... »
- Si on veut installer l'un de ces paquets, il y a une commande pour cela (voir plus loin)
- Debian regarde aussi s'il y a eu des améliorations des paquets qui sont déjà installés chez vous

Configuration des dépôts

- La liste des dépôts est placée dans le fichier `/etc/apt/sources.list`
- Voici un exemple (**mais pas réel !**) :

```
# commentaire
deb http://archive.debian.org/debian jessie main
deb http://archive.debian.org/debian jessie contrib
deb http://mirror.aarnet.edu/debian jessie non-free
```

Syntaxe de sources.list

- Chaque ligne de `/etc/apt/sources.list` est composée de :
 - Le mot clé `deb` (pour les binaires) ou `deb-src` (pour les paquets sources)
 - L'URL du dépôt
 - La version de la distribution : `wheezy`, `jessie`, `sid`...
 - Chaque version Debian a un petit nom de code
 - Les listes à considérer : `main`, `contrib`, `non-free`...
 - Ce sont différents groupes avec des licences différentes
 - Selon les Debian, ces listes ont des noms différents

Ajout d'un dépôt (Ubuntu)

- Dans certains cas, on peut vouloir ajouter un dépôt spécifique :
 - Bénéficiaire de paquets mis à jour plus fréquemment, ex : openshot dernière version
 - Bénéficiaire d'un logiciel qui n'est pas standard, ex : tor
- En général, c'est un dépôt qui ne contient qu'un seul paquet
 - On les appelle des Private Package Archive (PPA)

Ajout d'un PPA

- Ces dépôts ont un nom du type `ppa:identifiant`
- Il y a une commande pour les ajouter :
`sudo add-apt-repository ppa:identifiant`
- Exemple pour avoir blender dernière version :
`sudo add-apt-repository ppa:irie/blender`
- Il y a la vérification du dépôt (pour ne pas introduire de virus) par une signature cryptée

Suppression d'un PPA

- Les PPA sont ajoutés à votre système sous la forme d'un sous-dossier dans :

```
/etc/apt/sources.list.d
```
- Pour enlever un PPA de votre système, il faut seulement supprimer le dossier à son nom puis mettre à jour les paquets.

Mise à jour du catalogue local

- Lorsqu'on change la liste des dépôts, il faut reconstruire le catalogue des paquets de votre machine :
 - Savoir ce qu'il y a comme paquets dans les dépôts
 - Savoir ce que vous avez déjà comme paquets
- La commande est :
`sudo apt-get update`

Elle contacte les dépôts et télécharge leur liste

Liste des paquets des dépôts

- La commande `apt-cache pkgnames` affiche la liste de tous les paquets connus (mais pas forcément installés chez vous)
- C'est une liste très longue. On peut la limiter en mettant un préfixe en paramètre, exemple :

```
apt-cache pkgnames libgnome
```

liste tous les paquets connus dont le nom commence par ce mot (ex : libgnome = librairie gnome, l'interface graphique de Debian)

Recherche d'un paquet

- La commande `apt-cache search expr` cherche les paquets qui sont relatifs à l'expression régulière (type grep) fournie :

```
apt-cache search '[dD]oom'
```
- On obtient la liste des noms des paquets
 - Mais parfois, le mot qu'on cherche est seulement dans la description (liste très longue si mal choisi)

Recherche sur un fichier

- La commande `dpkg-query --search fichier` cherche le paquet installé qui contient le fichier indiqué
 - Ex : `dpkg-query --search /usr/bin/iconv`
- Ça ne marche pas avec les paquets non installés, il faut alors aller, par exemple voir :
<http://packages.debian.org/stable/>

Liste des paquets installés

- La commande `dpkg-query --list` affiche la très longue liste des paquets installés ou ayant été installés sur votre système
 - Les premières lettres indiquent l'état du paquet : i = installé, u=pas installé. Lire la doc pour les détails.
 - On peut préciser le nom d'un paquet, ou une expression régulière :
`dpkg-query --list 'vi*'`

Informations sur un paquet

- La commande `apt-cache show nom` affiche les informations sur le paquet indiqué

```
apt-cache show vavoom
```

- On obtient :
 - Version : numéro exact de réalisation
 - Depends : paquets qu'il faut d'abord installer avant celui-ci
 - Description : les infos sur le paquet

Détails d'un paquet

- La commande `apt-cache showpkg nom` affiche les détails techniques sur le paquet *nom*
`apt-cache showpkg genius-common`
- On obtient :
 - Version : numéro exact de réalisation
 - Dependencies : paquets qu'il faut d'abord installer avant celui-ci
 - Reverse depends : paquets qui ont besoin de celui-ci

Contenu d'un paquet installé

- La commande `dpkg-query --listfiles paquet` affiche la liste des fichiers d'un paquet
 - Ex : `dpkg-query --listfiles geany`



Installation d'un paquet

- La commande `sudo apt-get install nom` télécharge et installe le paquet *nom*
`sudo apt-get install genius`
 - Avant de le télécharger et de l'installer, la commande vérifie s'il a des dépendances non installées et les installe aussi récursivement
- Les paquets sont tous téléchargés dans `/var/cache/apt/archive` avant d'être installés, ce sont des fichiers `.deb`

Suppression d'un paquet

- La commande `sudo apt-get remove nom` désinstalle le paquet *nom*
 - L'option `--purge` supprime en plus ses fichiers de configuration (réinstallation => redépart de zéro)
`sudo apt-get remove --purge genius`
- Le fichier.deb reste dans `/var/cache/apt/archive`
- `sudo apt-get clean` supprime aussi le fichier.deb

La commande apt-get (résumé)

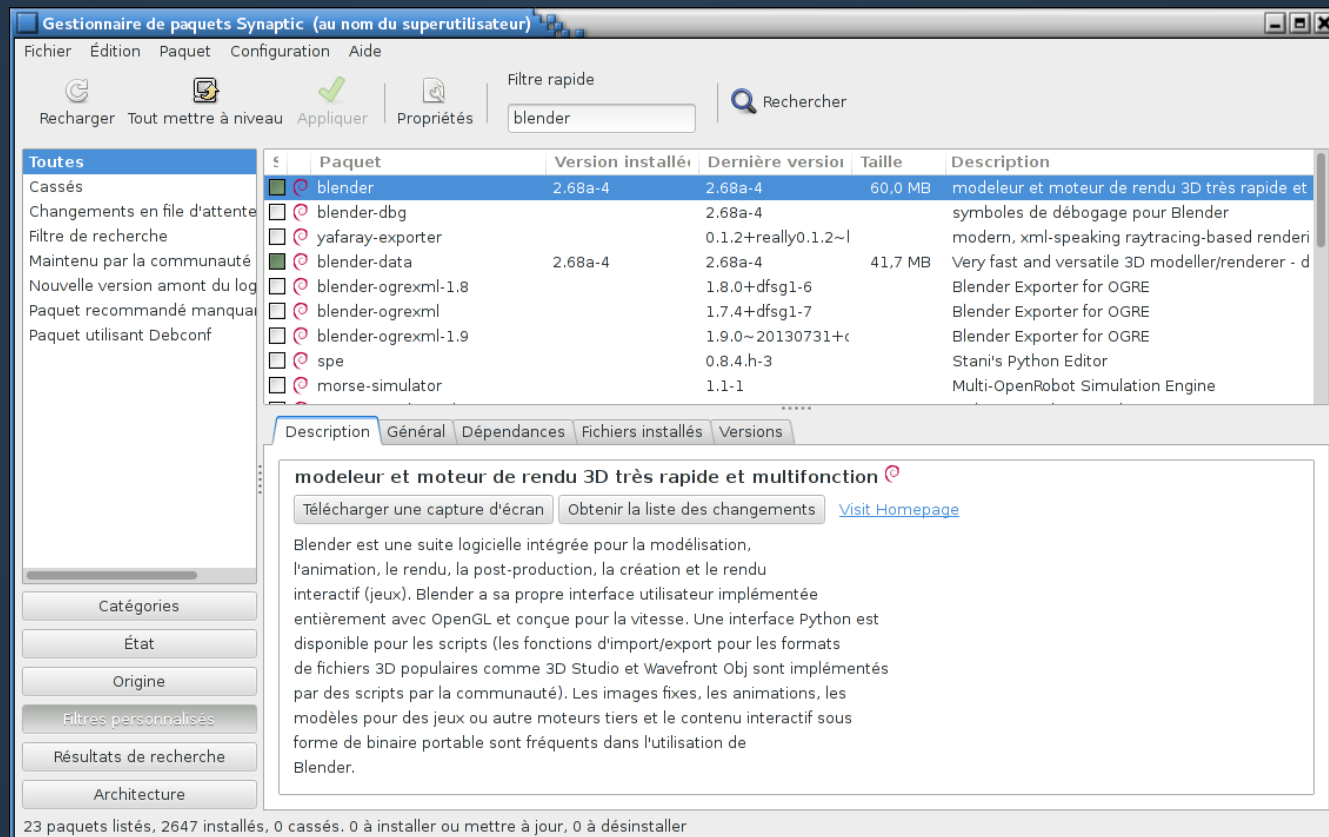
- Cette commande possède des variantes :
 - `update` : met à jour la liste des paquets
 - `install nom` : installe le paquet `nom`
 - `remove [--purge] nom` : supprime ce paquet
 - `upgrade` : met à jour les paquets installés
 - `dist-upgrade` : met à jour l'ensemble des paquets
 - `clean` : enlève les fichiers `.deb` téléchargés
 - `autoremove` : supprime les paquets devenus inutiles
- La commande **aptitude** est similaire

Tâches d'administration

- Installer un paquet :
 - 1) `sudo apt-get update`
 - 2) `sudo apt-get install nom`
- Mettre à jour le système (cf Windows Update) :
 - 1) `sudo apt-get update`
 - 2) `sudo apt-get upgrade` ou `dist-upgrade`
 - 3) `sudo apt-get autoremove`
 - 4) `sudo apt-get clean`


Interface graphique

- La commande apt-get existe avec une interface graphique plus sympathique : **synaptic**



10.3 – Création d'un paquet

Cette partie explique comment on crée un
paquet Debian



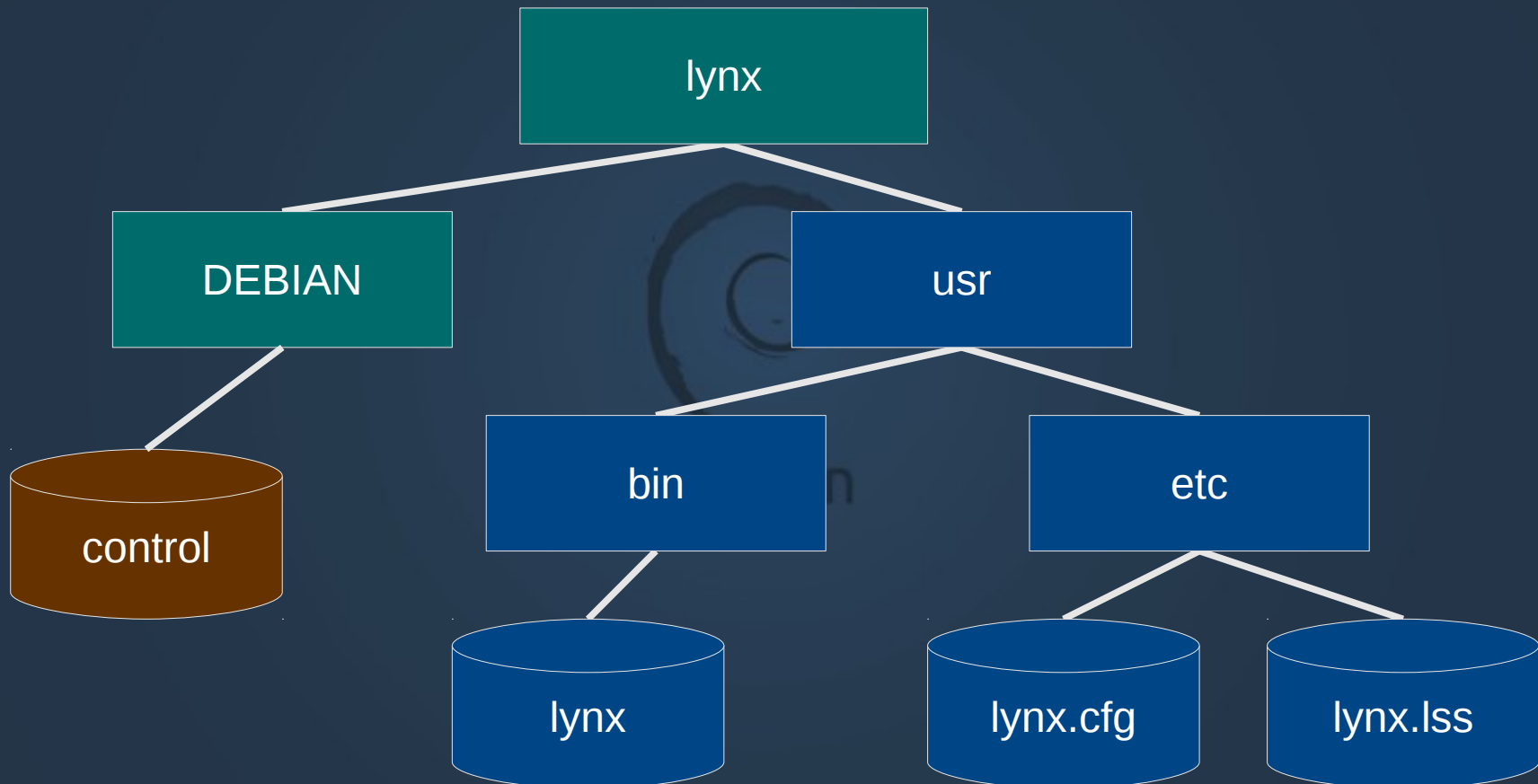
Debian

Structure d'un paquet debian

- Un fichier **.deb** est une archive contenant :
 - Un dossier **DEBIAN** contenant
 - Un fichier **control** qui décrit le paquet
 - Attention : rw seulement pour U, r pour les autres
 - Des dossiers qui ressemblent à l'arbre Unix et contenant les fichiers du paquet
 - Sous-dossier usr, usr/bin, usr/lib, etc, usr/share... selon les endroits où on veut mettre des fichiers
- Cette archive est créée par dpkg-deb

Exemple lynx.deb

- Voici l'arborescence (minimale) qu'il faut créer :



Le fichier DEBIAN/control

```
Package: lynx          nom du paquet
Version: 2.8.8        numéro de version
Section: web          catégorie
Priority: optional
Architecture: amd64   binaire 64 bits
Depends: libc6 (>= 2.8) dépendances
Maintainer: pierre <pierre@chezmoi>
Description: navigateur en mode texte
! mettre une ligne vide !
```

Création du paquet

- Il suffit ensuite de se placer au dessus de cette arborescence et de taper

```
dkpg-deb --build paquet
```

- Cela crée un fichier appelé *paquet.deb* contenant tout ce qu'il y a dans le répertoire *paquet*

- On peut afficher ce qu'il y a dedans avec :

```
dkpg-deb --contents paquet.deb
```

```
dkpg-deb --info paquet.deb
```


Installation, désinstallation

- Pour l'installer, ce n'est pas apt-get mais :

```
sudo dpkg --install paquet.deb
```

- Pour le désinstaller :

```
sudo dpkg --remove paquet
```

(sans l'extension .deb)

Debian

Remarques

- Il s'agit du cas le plus simple : un binaire, un fichier de configuration
- Dans un cas plus élaboré, on peut rajouter des scripts bash dans DEBIAN, en plus de control :
 - `preinst` : lancé avant d'installer le paquet
 - `postinst` : lancé juste après avoir installé le paquet
 - `pre rm` : lancé juste avant de supprimer le paquet
 - `post rm` : lancé juste après l'avoir supprimé
- Utilité : configurer le paquet, lancer un service...

Exemple de postinst

- Voici un exemple de ce qu'on peut mettre dans DEBIAN/postinst :

```
#!/bin/bash

chmod ugo+x /usr/bin/lynx
chmod ugo+r /usr/etc/lynx.*
```

- Ne pas oublier de le rendre exécutable
- Ce script est lancé après la copie des fichiers du paquet aux endroits prévus (/usr/bin...)