Master 2 d'Ingénierie Informatique
de l'Université Paris VII - Denis Diderot

Rapport de projet long

Juppix
(Jussieu Gnu/Linux Live-CD)

Glenn ROLLAND
<*glenux@fr.st*>

Sous la direction de

Juliusz Chroboczek
<*jch@pps.jussieu.fr*>
Chercheur au laboratoire PPS

Année 2005-2006

Table des matières

1 Introduction

1.1 Présentation du projet

Juppix est un projet visant à créer une distribution Live-CD GNU/Linux, c'est à dire un système d'exploitation complet sur CD-ROM, utilisable sans installation sur le disque dur.

Il s'agit d'une version du Live-CD GNU/Linux Knoppix à laquelle ont été ajoutés les logiciels utilisés par les étudiants de premier cycle d'informatique à l'Université Paris 7. On y trouve notamment :
– le kit de développement Java de Sun 1.5.0 de Sun ;
– la classe Java "Deug" utilisée en IF1 ;
– un environnement de développement complet pour O'Caml.
– un serveur Web Apache 2 avec PHP5
– quelques utilitaires (GnuPG, Darcs, etc.) en plus

Par contre l'interface utilisateur (KDE), l'environnement de développement pour le C (Emacs, Gcc et Gdb) ainsi que l'environnement TeX (TeTeX) sont inchangés par rapport à ceux de la Knoppix.

1.2 Remerciements

C'est avec plaisir que j'adresse mes remerciements aux personnes qui ont contribué à faire de projet long une expérience enrichissante et, en particulier, à :
– Juliusz Chroboczek, pour m'avoir fait confiance pour la réalisation de ce projet, pour ses conseils ainsi que pour sa disponibilité ;
– Sylvain Lebresne, pour sa participation active et ses conseils avisés pour ce projet.

2 Fonctionnement

2.1 Téléchargement

La première chose à faire pour utiliser Juppix est de se la procurer.

A l'heure actuelle, le CD-ROM n'est disponique qu'aux étudiants de Licence d'Informatique à l'Université Paris 7. Par contre les sources du projet sont disponibles en ligne, et téléchargeables avec `darcs` de la façon suivante :

```
$ darcs get http://www.pps.jussieu.fr/~jch/software/repos/juppix
```

2.2 Arborescence

Une fois téléchargées, les sources de Juppix se présentent sous la forme d'un ensemble de scripts shell, de fichiers de données et de divers répertoires. On y remarquera particulièrement les scripts suivants :

`"cleanup.sh"` **:** supprime les fichiers résiduels et dé-monte les systèmes de fichiers utilisés uniquement pendant la génération du live-cd,

`"makejuppix.sh"` **:** génère l'image ISO-9660 de la Juppix,

`"makejuppix2.sh"` **:** participe au processus de génération de la Juppix et qui est appelé par `"makejuppix.sh"`.

De même sont présent les fichiers ci-dessous :

`"dot.makejuppix"` **:** fichier de configuration pour la génération du live-cd,

`"bookmarks.txt"` **:** marques-pages pour les navigateurs web,

`"background.jpg"` **:** image de fond à utiliser pour le bureau et l'écran d'accueil.

Enfin, on y trouve aussi les répertoires suivants :

`"archives/"` **:** dépôt de paquetages `".deb"` supplémentaires, spécifiques à Juppix,

`"chroot-apt-cache/"` **:** dépôt des paquetages `".deb"`, sauvegardé d'une Juppix à l'autre, pour éviter de tout re-télécharger lors des mises-à-jour,

`"modified-files/"` **:** fichiers de configuration ou fichiers modifiés spécifiques à Juppix,

`"test-suite/"` **:** suite de test au run-time pour le live-CD.

2.3 Configuration préalable

A l'heure actuelle, la fabrication du CD-ROM Juppix necessite l'utilisation d'un système Debian installé (ou même dans un *chroot*) sur une architecture de type `x86`.

D'autre part, la construction d'une image du CD-ROM Juppix nécessite, au préalable, la copie de certains paquetages ou l'installation de modules non-inclus dans la distribution officielle de Debian.

2.3.1 Les paquetages Java

Création des paquetages sun-j2sdk

Commencez par télécharger les fichier binaire exécutable du kit de développement logiciel (J2SDK) et de l'environnement d'exécution (J2RE) Java de Sun sur le site `http://java.sun.com`. Les noms des fichiers en question devraient être de la forme : `"j2sdk-1_5_x_y-linux-i586.bin"` et `"j2re-1_5_x_y-linux-i586.bin"`

Installez le paquetage `"java-package"` :

```
# apt-get install java-package
```

Puis installez le logiciel `"fakeroot"`, qui permettra d'éliminer les effets de bords génants sur votre système lors de l'exécution des binaires exécutables de Sun lors de la création des fichier `"*.deb"` :

```
# apt-get install fakeroot
```

Lancez ensuite la création des paquetages Debian à partir des fichiers binaire :

```
# fakeroot make-jpkg j2sdk-1_5_x_y-linux-i586.bin
# fakeroot make-jpkg j2re-1_5_x_y-linux-i586.bin
```

Copiez ensuite les fichiers `"*.deb"` fraîchement créés par la commande précédente dans le répertoire `"archives"` des sources de Juppix.

Télécharger les classes DEUG

TODO: Indiquer où se procurer Deug_0.1*.deb.
Y-a-t'il un site officiel pour ces fichiers ?

Installation de cloop

Commencez par installer les packages Debian pour le module `"cloop"` et l'utilitaire `"create_compressed_fs"` :

```
# apt-get install cloop-utils cloop-src
```

Rendez-vous ensuite dans `/usr/src` et tapez :

```
# tar -xjvf cloop.tar.bz2
```

Rendez-vous ensuite dans `/usr/src/modules/cloop/debian` et tapez :

```
# m-a a-i cloop
```

2.4 Construire une image du Live-CD à partir des sources

Bien que les étapes de la génération d'une image de CD-ROM soient nombreuses et complexes, , la construction de la Juppix se resume pour l'utilisateur à un processus relativement simple, puis qu'il lui suffit de taper dans un terminal la ligne :

```
$ sudo sh ./makejuppix.sh
```

et de patienter un peu (pendant que le script exécute toute les étapes précédentes) pour construire une nouvelle impour construire une nouvelle "image" du CD-ROM Juppix. Il suffit ensuite de graver l'image ISO-9660 générée pour avoir une version utilisable.

Enfin, il est souhaitable de nettoyer les fichiers temporaires en tapant

```
$ sudo sh ./cleanup.sh
```

2.5 Utiliser une juppix

Une fois le CD-ROM juppix entre vos mains, il suffit de faire démarrer l'ordinateur dessus, en changeant si nécessaire les réglages du BIOS.

3 Description détaillée

3.1 Les étapes de la construction de la Juppix

Les grandes lignes de la construction de la Juppix sont les suivantes :

1. chargement de la configuration de l'utilisateur,
2. création des répertoires nécessaires à la construction du système ,
3. montage et copie du système de la Knoppix dans un répertoire de travail,
4. copie des fichiers de configuration modifiés dans le répertoire de travail,
5. création d'un chroot dans le répertoire de travail,
6. désinstallation des logiciels inutiles,
7. mise à jour du système,
8. installation des paquetages de la juppix,
9. remplacement/modification des fichiers de configuration par ceux fournis par la Juppix,
10. activation/désactivation des services (Apache/MySQL) au démarrage du système,
11. construction de l'image compressée (cloop),
12. construction de l'image ISO-9660 finale.

Certains de ces points, utiles à la "personalisation" de la Juppix et qui ne relèvent pas que de la technique sont détaillés ci-après.

3.2 Configuration spécifique à l'utilisateur

Le processus de génération de l'image du CD-ROM de la Juppix doit être adapté à votre système.

Le fichier de configuration `"dot.makejuppix"`, une fois copié à la racine de votre répertoire personnel sous le nom `".makejuppix"` permet d'adapter certaines des variables des scripts de Juppix à votre système.

A l'heure actuelle, le fichier de configuration est le suivant :

```
## repertoire ou se trouvent les scripts de Juppix
prefix="/home/user/projets/juppix/juppix"

## chemin du fichier de l'image ISO-9660 de la Knoppix
## qui sera utilisee comme base
image="$prefix/../knoppix/KNOPPIX_V4.0.2CD-2005-09-23-EN.iso"

## chemin du repertoire qui sera utilisee pour monter
## l'image ISO-9660 precedente
mntdir="$prefix/../mnt"

## chemin du fichier de l'image compressee (cloop) contenue
## dans l'image ISO-9960 de Knoppix
image2="$mntdir/KNOPPIX/KNOPPIX"

## chemin du repertoire qui sera utilise pour monter
## l'image compressee (cloop) precedente
mntdir2="$prefix/../mnt2"

## numero de version de la Knoppix utilisée comme base.
knoppix_version="4.0.2"
```

3.3 Système de génération de bookmarks

Le script "`bookmark-gen.sh`" sert à générer les fichiers contenant les marques pages affichés dans les navigateurs web Mozilla Firefox et Konqueror. Ces marques-pages sont respectivement au format HTML et XML XBEL et sont produits à partir d'un fichier texte contenant sous forme structurée des répertoires et des marques-pages.

Dans ce fichier d'entrée, les répertoires sont définis "entre parenthèses". Les noms des répertoires doivent être écrits juste après la parenthèse ouvrante, et les marques-pages qui y sont inclus doivent figurer avant la parenthèse fermante. Un répertoire peut lui même contenir d'autres répertoires. De plus, il peut y avoir qu'une unique définition (répertoire ou marque-page) par ligne. Enfin, les parenthèses fermantes doivent aussi figurer seules sur une ligne.

Exemple de fichier d'entrée.

```
( Demo
  ( Divers
    http://thedailywtf.com/              "The daily WTF"
    http://www.dina.kvl.dk/~abraham/religion/   "The Church of Emacs"
    http://livre.point6.net/             "IPv6: théorie et pratique"
  )
  ( Cours
    http://.../Emplois_du_temps.html  "Emplois du temps"
    http://.../~ rifflet/enseignements/IF1/  "IF1"
```

```
    )
)
```

Exemple de fichier généré (bookmarks équivalents pour Konqueror).

```
<!DOCTYPE xbel >
<xbel>
  <folder folded="no">
    <title>Demo</title>
        <folder folded="no">
          <title>Divers</title>
          <bookmark href="http://thedailywtf.com/" >
            <title>The daily WTF</title>
          </bookmark>
          <bookmark href="http://www.dina.kvl.dk/~abraham/religion/" >
            <title>The Church of Emacs</title>
          </bookmark>
          <bookmark href="http://livre.point6.net/" >
            <title>IPv6: théorie et pratique</title>
          </bookmark>
        </folder>
        <folder folded="no">
          <title>Cours</title>
          <bookmark href="http://.../Emplois_du_temps.html" >
            <title>Emplois du temps</title>
          </bookmark>
          <bookmark href="http://.../~ rifflet/enseignements/IF1/" >
            <title>IF1</title>
          </bookmark>
        </folder>
  </folder>
</xbel>
```

3.4 Paquetages logiciels et répertoire de cache

Pendant le processus de génération du contenu de Juppix, les scripts suppriment puis (après une mise à jour) installent de paquetages logiciels, téléchargés d'un dépôt Debian sur internet.

Ces listes de paquetages logiciels à supprimer et installer sont définies dans le script `"makejuppix2.sh"`, respectivement dans les variables `"packages_removed"` et `"packages_added"`, comme en témoigne l'extrait du fichier ci-après :

```
packages_removed="kde-i18n-de kde-i18n-ja kde-i18n-pl kde-i18n-tr [...]\
    isdnutils-base isdnutils-xtools isdnvboxclient trans-de-en "

packages_added="ocaml-base ocaml-base-nox ocaml-compiler-libs [...]\
    manpages-posix manpages-posix-dev azureus amule amule-utils"

[...]
```

```
echo "Munging package collection..."
apt-get remove --purge --yes --force-yes $packages_removed
apt-get update || true
[...]
apt-get install --yes --force-yes $packages_added
```

Comme il est fréquent, lorsque l'on apporte des modifications à Juppix, d'avoir à générer plusieurs fois l'image du CD-ROM, il devient préférable d'éviter le téléchargement à l'identique des paquetages logiciels d'une génération à l'autre.

Le répertoire `"chroot-apt-cache"` sert justement à éviter cela. Avant d'installer les paquetages, il est attaché au répertoire dans lequel seront réceptionnés les fichiers `".deb"`, et il est détaché juste après l'installation des logicieils.

On notera toutefois que l'espace disque occupé par ce répertoire devient très vite non-négligeable, du fait de l'accumulation des versions successives des paquetages logiciels, si l'on travaille sur une longue période de temps sans le vider ou si l'on passe d'une version Knoppix à une autre comme base du live-cd Juppix.

3.5 Suite automatique de tests

La construction automatisée d'un système même si elle arrive à jusqu'à son terme, n'est pas forcément le signe que le système résultant est exempt de défauts.

Dans le cas de Juppix, on peut par exemple citer les points critiques suivants :
– les mises à jour de paquets vers une nouvelle version non-testée
– les modifications dans les scripts de construction `makejuppix*.sh`
qui tout deux peuvent amener à un système partiellement fonctionnel, instable ou totalement inutilisable, et qui ne répond donc plus aux besoins formalisés initialement.

Le besoin s'est donc rapidement fait sentir de prévoir les différents cas d'utilisations de la Juppix, et de vérifier chacun d'entre eux par des scripts exécutant les outils logiciels comme l'utilisateur, testant les codes de retour, cherchant les paquetages installés, ainsi que leur configuration respective.

A l'heure actuelle, la suite de tests vérifie les points suivants :
– l'installation des paquetages logiciels (apache2, mysql-server, gcc, g++, ocaml, php5, darcs),
– la configuration d'Emacs,
– la configuration et le fonctionnement du serveur HTTP Apache 2 (publication de HTML),

– le fonctionnement du serveur de bases de données MySQL (test fournis par MySQL),
– le fonctionnement de PHP5 (ligne de commande et module Apache2),
– l'installation du SDK Java de Sun et le fonctionnement des classes DEUG,
– le fonctionnement du compilateur O'Caml,
– le fonctionnement des compilateurs GCC et G++,

Cette suite de tests est conçue pour être lancée à partir d'un Live-CD Juppix en fonctionnement. Pour cela, procurez-vous les sources de Juppix et rendez-vous dans le répertoire principal de ces sources, puis tapez :

```
$ cd test-suite
$ sh testjuppix.sh
```

Si l'exécution des tests échoue avant la ligne :

```
TestJuppix: All scripts passed successfully !
```

alors votre Live-CD contient probablement une erreur concernant le sujet de la dernière ligne de texte qui fut affichée... et il faut recommencer la construction de la Juppix en ayant préalablement corrigé le point indiqué.

3.5.1 Fonctionnement des tests

Les scripts de tests doivent être placés dans le répertoire `"test-suite/scripts"`. Ensuite, le fichier `"test-suite/testjuppix.sh"` doit être modifié à l'emplacement opportun pour y placer l'exécution du nouveau fichier de tests, de la même façon que dans l'extrait suivant :

```
#!/bin/bash

TESTDIR=\pwd`
SCRIPTDIR=$TESTDIR/scripts
DATADIR=$TESTDIR/data

##
# -e: Exit immediately if a simple command exits with a non-zero status.
# -u: Prevent unbound variable errors

set -ue

##
# first test: package installation

. $SCRIPTDIR/packages.sh
```

```
##
# then test emacs

. $SCRIPTDIR/emacs.sh

# [...]
```

3.5.2 Structure d'un fichier de tests

Tous les scripts de la suite de test doivent posséder la structure suivante :

```
#!/bin/bash

set -ue

##
# déclaration des fonctions si le script est vraiment long
#

[...]

##
# exécution du code et appels des fonction définies plus haut

[...]
```

Dans les tests de Juppix, on préférera clarifier au maximum la lecture par l'écriture de scripts linéaires, et en ne définissant de fonctions que lorsqu'elle sont réellement nécessaires.

D'autre part, on prendra particulièrement soin lors des fonctions, de préfixer leurs noms avec le celui du fichier de test, afin d'éviter de redéfinir des commandes ou fonctions appelées par le script principal.

Par exemple, pour le test `"packages.sh"` on aura :

```
# exécution
packages_test() {
    local pkg=$1
    # test debian package
    echo -n "Packages: checking '$pkg' package... "
    local score=0
    if dpkg -l | grep -q "ii  $pkg "; then
        score=$(( $score + 1 ))
    fi
    if [ $score -gt 0 ]; then
        echo "done"
        return 0
    else
        echo "fail"
        echo "Error: Package '$pkg' is missing..."
```

```
        return 1
    fi
}

packages_test darcs
packages_test konsole
packages_test apache2
packages_test emacs21
packages_test gcc
packages_test g++
packages_test mysql-server
packages_test ocaml-base
packages_test php5 || packages_test php4
packages_test php5-mysql || packages_test php4-mysql
```