Les outils de développement OpenOffice.org (SDK)

Sommaire

Présentation

Ce document, élaboré dans le cadre des activités du LUG toulousain « CULTE » www.culte.org se propose de donner quelques informations pour l'utilisation du Kit de Développement d'OpenOffice.org. Après une présentation succincte du produit, nous aborderons l'interface de programmation, le modèle de composant UNO qui en constitue l'ossature. Nous apprendrons comment s'y retrouver dans la luxuriante documentation. Enfin, nous écrirons un petit programme qui ouvre le traitement de texte et y écrit quelques lignes. Ce programme sera développé dans l'environnement libre GNU/Linux, Eclise, gcj. Nous verrons ensuite comment en extraire un paquetage portable utilisable dans d'autres environnements.

OpenOffice

OpenOffice.org http://www.openoffice.org est issu d'une suite intégrée « WYSIWYG » développée en 1994 à partir d'une bibliothèque C++ portable par la société StarDivision (Marco Börries). Cette société a été rachetée en 1999 par Sun Microsystems. En juillet 2000, Sun publie le code sous licence SISSL puis en 2005, le code de la version 2.0 d'OpenOffice.org en LGPL. La version commerciale, qui se nomme toujours StarOffice http://www.sun.com/software/star/staroffice/index.jsp se distingue de la version libre (OpenOffice.org) par des ajouts propriétaires : base de données Adabas B, polices de caractères, dictionnaires, modèles, clip arts, filtres de conversion de fichiers, meilleur support des langues asiatiques, documentation professionnelle et support technique...

La suite comporte un traitement de texte, un tableur, un outil de présentation, un outil de dessin vectoriel, une interface de bases de données et un éditeur de formules mathématiques.

Capable de lire et d'écrire des documents de plusieurs formats Microsoft Office, c'est la suite bureautique fonctionnant aujourd'hui sur le plus grand nombre de plates-formes.

Le format natif de ses documents repose sur une définition XML ouverte publiée par l'OASIS nommé « Open Document »

http://www.oasis-open.org/specs/index.php#opendocumentv1.0

La version actuelle (29-01-2006) sous Linux est la 2.0.1. Il existe en ligne une bonne présentation à l'adresse : http://fr.wikipedia.org/wiki/OpenOffice.org . On y trouve également de nombreux liens très intéressants.

OpenOffice et Java

Le « coeur » d'OpenOffice.org est écrit en C++. Début 2005, lorsque sont apparues les premières moutures de ce qui allait devenir la 2.0, des voix (dont celle de R.M.Stallman) se sont émues en constatant la présence de plus en plus importante de code Java dans le projet. Pour mémoire, si les emplois de Java sont marginaux dans les versions antérieures à la 2.0, cette dernière requiert Java pour une grande partie des traitements XML (filtres), pour certains scripts, pour la base de données HSQLDB et dans certains rendus graphiques et des sons. Activer OpenOffice.org sans Java, ce qui reste encore possible, le prive de nombreux usages très intéressants. Le problème est que la licence Java n'en fait pas un logiciel libre et qu'il n'existe pas de version du Java de Sun pour de nombreuses plates-formes sur lesquelles tournent des versions de GNU/Linux ou des différents xBSD. Pour éviter une divergence des versions (un « fork ») préjudiciable à l'ensemble de la communauté, un accord a été conclu en mai 2005 entre Sun et la FSF. Il s'est appuyé sur les efforts des développeurs de GCJ (en particulier Coalan McNamara) pour permettre la livraison

d'une version d'OpenOffice.org compilée sans utiliser le compilateur propriétaire. C'est le cas en particulier des environnements Red-Hat , Fedora et Debian, Ubuntu. Constatant que la mise en oeuvre de la suite bureautique n'avait plus besoin d'aucun logiciel propriétaire, Stallman et la FSF ont stoppé leurs appels aux développeurs pour construire une version débarrassée de Java.

Http://www.fsf.org/news/open-office-java.html

Remontons aux sources

Le code source d'OpenOffice.org contiendrait environ 8 millions de lignes de code. Une chose est sûre : la distribution OOo actuellement disponible http://download.openoffice.org/2.01/source.html contient un répertoire nommé OOA680_m1 de 1,15 Go contenant 90059 fichiers répartis dans 11739 dossiers. La plus grande partie du code est en C++. La compilation et les tests nécessitent des compilateurs ou interpréteurs C++, Java, Perl, Python, M4. Les soucis de portabilité rendent le code difficile à lire mais font d'OOo la seule suite bureautique disponible sur un grand nombre de plates-formes. Il existe des distributions officielles pour MS Windows (98, ME, 2000, XP, 2003), Solaris Sparc, Solaris x86, Linux (Gnome), Mac Os X (11).et de nombreux autres portages plus ou moins aboutis (xBSD , Linux KDE, Mac Os X sans X11 -Neooffice-, sans parler des différentes compilations des distributeurs Linux évoquées plus haut...). Certains outils de développement largement disponibles ont été récrits pour faciliter les portages; c'est le cas de l'utilitaire « make » avec « dmake ».

Les logiciels libres intégrés dans les sources

La suite dépend de nombreux logiciels libres (GPL ou autres) disponibles par ailleurs. Suivant les distributions et le niveau d'intégration de la compilation distribuée, on trouve dans la distribution d'OpenOffice.org, des versions (modifiées ou non,) de :

- Spirite (analyseur syntaxique récursif en C++),
- Libcurl (Liaison avec les serveurs Internet en C),
- epm (outil de portabilité Unix en C),
- expat (analyseur XML en C),
- freetype (gestion des polices en C),
- icu (outils unicode en C++),
- une bibliothèque Jpeg (en C),
- libwpd (pour l'importation de documents WordPerfect en C++),
- libxml2 (un autre analyseur portable XML en C),

- libxmlsec (en C),
- msfontextract (pour la compression des fichiers microsoft .cab .hlp ...),
- neon (WebDAV en C),
- np_sdk (Boite à outils Netscape en C++),
- portaudio (une plate-forme audio portable en C++),
- Python (en C),
- Sablotron (un processeur XSL en C++),
- libsndfile (une librairie audio en C),
- zlib (librairie portable de gestion des fichiers compressés en C).

Et les outils Java :

- accessibility (ensemble d'outils d'accessibilité (ex équipements Braille...)
- Beanshell (script)
- hsqldb (gestionnaire de base de données)
- Rhino (javascript)
- Xalan (le processeur XSL d'Apache)

L'avantage de cette pratique est d'éviter les problèmes liés à l'incompatibilité des versions, l'inconvénient réside dans le risque de se retrouver avec plusieurs versions d'un même programme sur une même machine avec des effets de bord inévitables. Les intégrateurs des grandes distributions Linux tentent de limiter ce genre de problème. Par exemple sous Fedora Core 4, OpenOffice.org utilise le seul paquetage Python installé. Ce n'est pas le cas sous MS Windows.

Compte tenu de leur complexité, peu de gens se lancent dans la compilation et l'installation à partir des sources (même sous Gentoo...). Heureusement, tout cela n'est pas nécessaire pour étendre les possibilités d'OOo et l'adapter à nos besoins. OOo dispose d'une interface de programmation (API) qui permet de programmer des applications en utilisant OOo.

L'interface de programmation

Cette API repose sur la technologie « composant » d'Openoffice.org. C'est-à-dire sur la définition d'un grand nombre d'interfaces définies abstraitement dans le langage IDL (Interface Definition Langage) semblable à celui de CORBA. Tandis que la technologie « composant » définit la façon dont les différents composants ou applications communiquent entre eux et comment les programmes écrits dans différents

langages accèdent à l'API, cette dernière définit les interfaces pour accéder aux fonctionnalités d'OOo indépendamment des langages de programmation. (http://api.openoffice.org). Les concepteurs de l'API se sont donnés des objectifs très ambitieux en matière d'indépendance vis à vis des versions, à la durabilité, au réemploi et à l'indépendance vis à vis des langages de programmation (http://api.openoffice.org/common/designgoals.html). Cette API doit peu évoluer et seulement par des ajouts.

UNO

La technologie « composant » d'OpenOffice.org se nomme UNO (Universal Network Object). Elle est issue du projet UDK (UNO Development Kit). UNO est un modèle de composant reposant sur des interfaces. Concrètement, les mécanismes d'UNO procurent au développeur des références à des objets qui, dans le pire des cas peuvent avoir été écrits dans d'autres langages et instanciés sur d'autres machines du réseau disposant éventuellement d'une autre architecture. http://udk.openoffice.org . UNO n'est pas la seule couche d'abstraction d'OpenOffice.org. Au niveau de l'infrastructure, on trouve aussi les couches d'objets graphiques et la couche d'abstraction SAL (System Abstraction Layer) qui garantit la portabilité du code C++). Mais ces autres couches, assez nombreuses à bien y regarder, peuvent être ignorées du programmeur d'extensions (sauf SAL s'il utilise C++).

Les langages de programmation qui possèdent des liens complets avec UNO sont

- C++ sur les plates-formes désignées en http://porting.openoffice.org.
- Java
- Python

On accède aussi aux composants UNO sans pouvoir en développer de nouveaux avec

- OpenOffice.org Basic
- BeanShell
- JavaScript
- Les environnements Microsoft CLI et OLE.

Comme dans la documentation, les principaux exemples seront fournis en Java. La couche d'abstraction SAL et les spécificités du langage (en particulier l'usage des « templates » plus ou moins comprises des différents compilateurs) rendent l'emploi de C++ plus difficile. La passerelle Python utilise celle de C++ et les fonctions disponibles, souvent analogues à celles du Basic facilitent la tâche du programmeur mais dissimulent le fonctionnement réel et n'aident pas à la compréhension du fonctionnement.

Que trouve-t-on dans UNO ?

Le coeur est constitué par le langage UNOIDL qui permet de définir les interfaces que doivent implémenter les composants pour être capables de fonctionner dans un environnement UNO. A partir de ces définitions, des outils sont disponibles qui génèrent les fichiers d'en-tête et les librairies pour les langages cibles.

L'instantiation d'un composant dans un contexte UNO se fait à travers le « gestionnaire de service ». Ce dernier a accès à une base de données des composants enregistrés et est capable de les créer à l'appel de leurs noms.

Les spécifications UNO définissent :

- Un certain nombre de *types* simples (char, long, float, string...). L'un des rôles des passerelles est de lier ces types aux types des langages cibles (C++, Java, Python). Il existe un type « Any » qui les représente tous.
- Les valeurs prédéfinies (const, enum)
- Les *structures* définissent des enregistrements composés de plusieurs éléments. Il n'y a pas d'héritage multiple mais un polymorphisme analogue aux « templates » de C++ est proposé.
- Les *séquences* sont des ensembles d'éléments de même type.
- Les *propriétés* sont des données accessibles par leur nom à travers une interface générique -GetPropertyValue() et SetPropertyValue()-.
- Les *singletons* désignent les objets nommés dont une seule instance est autorisée dans le contexte d'un composant UNO.
- Les *exceptions* définissent le mécanisme de traitement des erreurs à l'exécution.
- Les *interfaces* définissent les aspects extérieurs particuliers d'un objet auxquels on accède sans rien savoir de leur représentation interne. Pour celui qui les implémente : ce sont des spécifications abstraites.
- Les *services* ont été modifiés profondément à partir de la version 2.0 avec l'introduction de l'héritage multiple des interfaces. Dans la dernière version, un service spécifie un objet qui implémente une seule interface. Une requête à un gestionnaire de service l'intantie à l'appel du nom du service. Un tel service n'existe que dans un contexte de *composant*. Mais les développeurs ont aussi hérité et conservé pour des raisons de compatibilité de très nombreux et très utilisés services qui fonctionnent sur un autre mode. Ces services comportent de nombreuses interfaces, héritent d'autres services, possèdent parfois leurs propres données qui ne sont pas traitées comme des *propriétés.* Ils ne propagent pas leur contexte.
- Les *modules* sont des espaces de noms (les « namespaces » de C++ ou les

« packages » de Java). Ils contiennent toutes sortes d'objets UNO regroupés en blocs qui structurent l'API d'un point de vue logique. Ils ne sont pas structurés en fonction des différentes applications d'OpenOffice.org (Writer, Calc, Draw...) mais selon des catégories fonctionnelles souvent communes à toutes ces applications.

- Un *composant* est défini par un gestionnaire de service et un certain nombre de données associées (un contexte). Il regroupe des objets UNO dans une bibliothèque (code source compilé et lié). Il rassemble tout un aspect de l'application (le traitement de texte, le tableur... sont des composants).

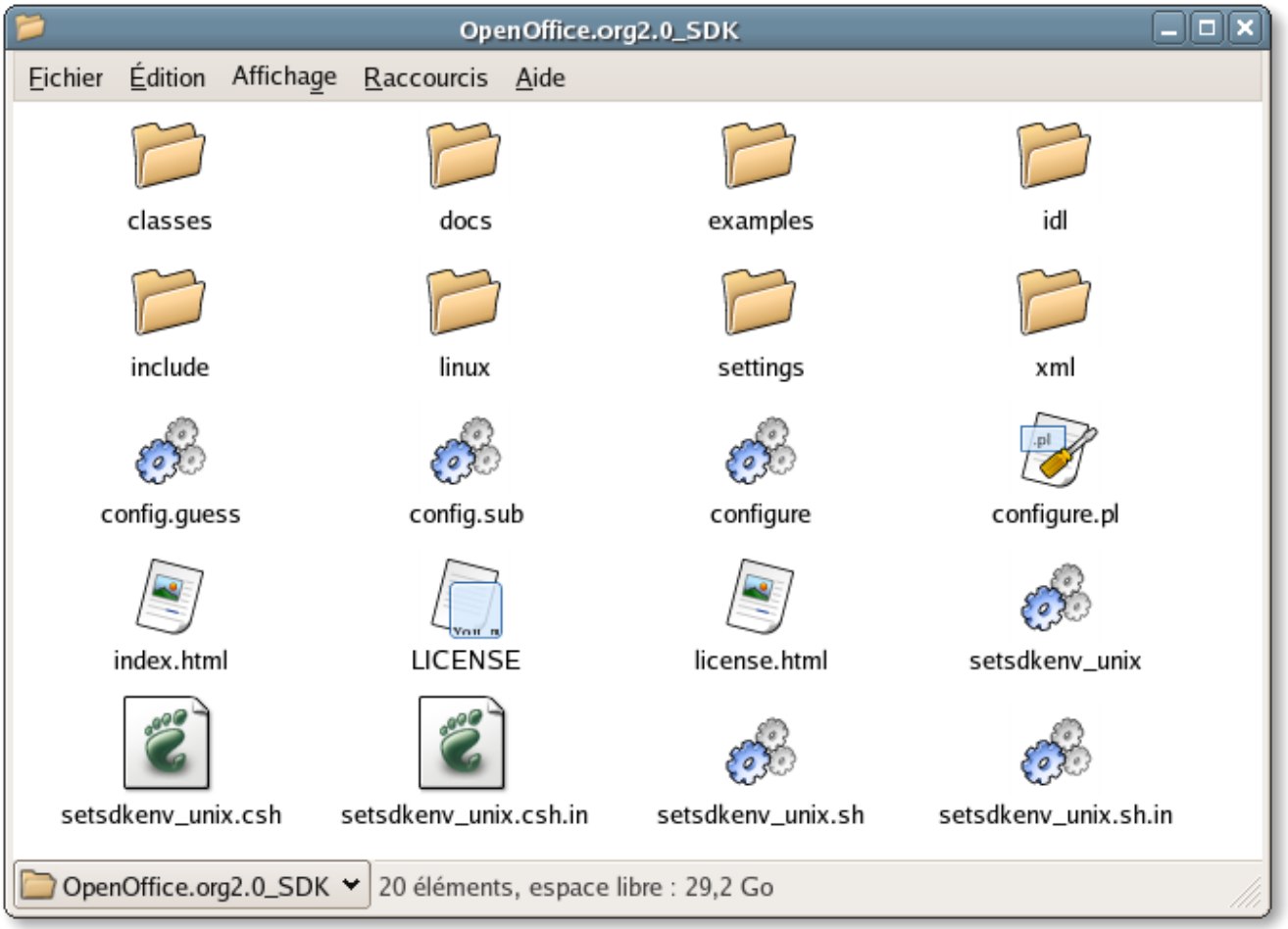
Le kit de développement

Il est téléchargeable à l'adresse : http://download.openoffice.org/680/sdk.html

On obtient une centaine de Mo dont les trois quarts sont constitués par la documentation. Cette dernière est également accessible à l'adresse : http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html

Le reste se répartit en :

- les fichiers d'en-tête pour les programmes C et C++.
- les classes pour le chargement des programmes Java.
- Les définitions en UNOIDL des différents modules, services et interfaces
- Les déscriptions xml des différents modules
- les outils de développement décrits en http://udk.openoffice.org/common/man/tools.html



Pour un développement en Java qui ne construit pas d'autres composants que ceux qui existent déjà, seul le répertoire « classes » est nécessaire. Il contient les objets qui permettent de charger les objets UNO. Ces derniers sont déjà présents sous forme de programmes ou de librairies dans l'installation d'OpenOffice.org. A titre d'exemple, ils sont installés par la Fedora Core 4 en « /usr/lib/openoffice.org2.0/program ».

La documentation

La documentation en ligne est très abondante, mais il faut un certains temps d'adaptation avant de s'y retrouver. Cette documentation est foisonnante. Ne ronchonnez pas trop après elle, il manque des choses mais vous allez vite vous apercevoir très souvent que ce que vous avez mis si longtemps à comprendre était parfaitement expliqué à un endroit auquel vous n'aviez pas fait attention. Observez les listes de discussion, s'y expriment plein de gens qui connaissent bien OOo, leur conversation est instructive. La documentation dont je parle est bien sûr en anglais. Il existe aussi, pour des raisons historiques beaucoup d'information en langue allemande. En français, nous avons encore des efforts à faire....

La documentation générale en anglais sur OOo, vous la trouverez en :

http://support.openoffice.org/index.html

En français tout est accessible à partir de là :

http://fr.openoffice.org/

Concernant l'api le point d'entrée est :

http://api.openoffice.org/SDK/index.html

On y trouve en particulier le très riche Developer's Guide :

http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html

Les définitions abstraites des modules, services et interfaces... dans l'UNO IDL:

http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html

Tout ce qui concerne les spécifications pour le dévelopement en C++ :

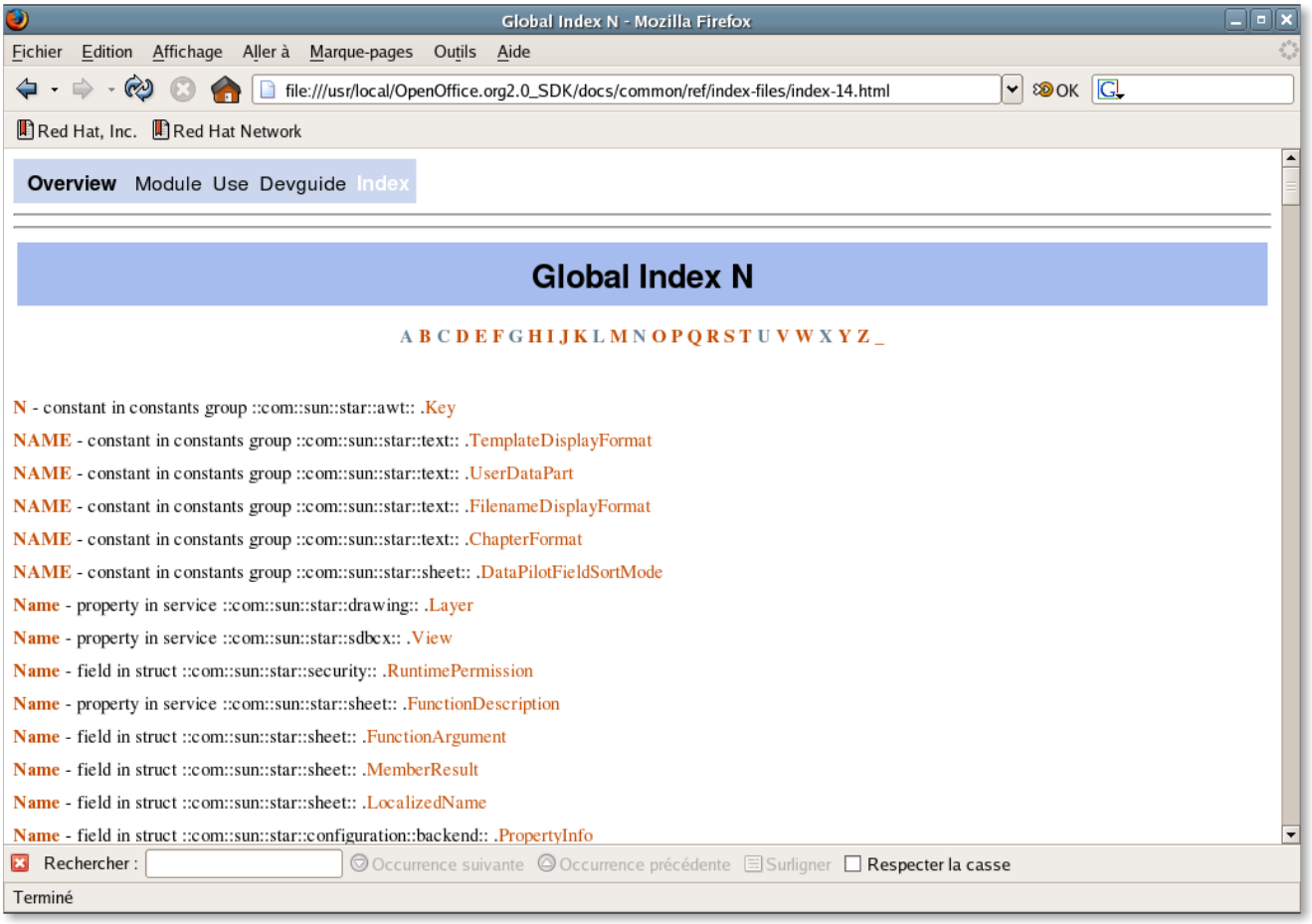
http://api.openoffice.org/docs/cpp/ref/names/index.html

Et en java :

http://api.openoffice.org/docs/java/ref/overview-summary.html

Les quatre dernières ressources sont distribuées avec le SDK. Vous les avez donc localement sur votre machine. L'accès est alors plus rapide.

Attention, à cause de la spécification « abstraite » des objets UNO, les définitions garantissent que si un service comporte une certaine interface, son implémentation met à disposition toutes les caractéristiques de cette interface. C'est une condition nécessaire, mais l'implémentation réelle peut tout aussi bien procurer d'autres objets non décrits. L'arborescence des objets décrits dans la documentation commence par les modules puis pour chaque module ses services (s'il y en a) et les interfaces concernées. Les pages des services documentent les interfaces et les grandeurs qu'ils implémentent. Les pages des interfaces spécifient les méthodes qui doivent être écrites par les programmeurs implémentant ces services. Pensez à utiliser les Index de ces documentations en ligne. Ils référencent par ordre alphabétique tous les types d'objets UNO. Attention : deux objets différents de deux modules différents peuvent porter le même nom !



Pour les hardcoders en C++ vous trouverez le très bon document de Serge Moutou http://perso.wanadoo.fr/moutou/ .

A noter qu'il existe aussi deux excellents livres sur la programmation d'OpenOffice.org en Basic:

En français :

> Programmation d'OpenOffice.org 2 ; Macros en Api / Laurent Godart, Bernard Marcelly.-- Eyrolles, 2005.-- 724 p., ISBN 2-212-11763-9

En anglais :

> OpenOffice.Org Macro Explained / Andrew Pitonyak.-- Hentzenwerke Publishing, 2004.-- 467 p., ISBN 1-930919-51-4

La plupart des programmes réellement utiles pour OpenOffice.org peuvent s'écrire très facilement en Basic. Mais Basic, parce qu'il facilite le travail du programmeur, dissimule nombre de concepts essentiels dont la compréhension est nécessaire pour bien manipuler l'API.

Un premier programme

Presque tous les exemples du SDK sont écrits en Java. L'installation documentée concerne une plate-forme avec le Sun Java Development Kit (1.4.2_05 ou supérieur) et l'interface de développement NetBeans. En suivant les instructions données au début du « Developer's Guide » on parvient facilement à compiler et exécuter un premier programme.

Notre exemple utilisera « Fedora Core 4 x86 » avec le « Gnu compiler for Java » http://gcc.gnu.org/java/index.html et l'interface de développement « Eclipse » http://www.eclipse.org/

Sur FC4 tous ces outils sont installés avec le « développement ».

La première chose à faire est de récupérer ou d'éditer un programme Java simple. Voici un exemple :

```
/**
 * Premier test d'utilisation d'une action sur une installation
distante
 *
 */
import com.sun.star.bridge.XUnoUrlResolver;
import com.sun.star.lang.XComponent;
import com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.uno.XComponentContext;
import com.sun.star.uno.UnoRuntime;
import com.sun.star.frame.XComponentLoader;
import com.sun.star.beans.PropertyValue;
import com.sun.star.beans.XPropertySet;
import com.sun.star.text.XText;
import com.sun.star.text.XTextRange;
import com.sun.star.text.XTextDocument;

public class documentLoader {

      /**
       * @param args
       */

      public static void main(String[] args) {
            String[] argsn = new String[] {
            "uno:socket,host=192.168.0.10,port=8100;urp;StarOffice.Se
rviceManager",
            "private:factory/swriter" };
            try {

①                XComponentContext xcomponentcontext =
com.sun.star.comp.helper.Bootstrap.createInitialComponentContext(null
);

②                XMultiComponentFactory xmulticomponentfactory =
xcomponentcontext.getServiceManager();
```

```
③                Object objectUrlResolver =
xmulticomponentfactory.createInstanceWithContext(
                                          "com.sun.star.bridge.UnoUrl
Resolver",
                                          xcomponentcontext);
④                XUnoUrlResolver xurlresolver =
(XUnoUrlResolver)UnoRuntime.queryInterface(XUnoUrlResolver.class,obje
ctUrlResolver);

⑤                Object objectInitial =
xurlresolver.resolve(argsn[0]);

⑥                xmulticomponentfactory =
(XMultiComponentFactory)
UnoRuntime.queryInterface(XMultiComponentFactory.class,
objectInitial);

⑦                XPropertySet xpropertysetMultiComponentFactory =
(XPropertySet)UnoRuntime.queryInterface(XPropertySet.class,xmulticomp
onentfactory);

⑧                Object objectDefaultContext =
xpropertysetMultiComponentFactory.getPropertyValue("DefaultContext");

⑨                xcomponentcontext =
(XComponentContext)UnoRuntime.queryInterface(XComponentContext.class,objectD
efaultContext);


⑩                Object xdesktop =
xmulticomponentfactory.createInstanceWithContext("com.sun.star.frame.
Desktop",

      xcomponentcontext));
⑪                XComponentLoader xcomponentloader =
(XComponentLoader) UnoRuntime.queryInterface(XComponentLoader.class,
xdesktop);


⑫                XComponent xcomponent =
 xcomponentloader.loadComponentFromURL(argsn[1],"_blank",0,new
PropertyValue[0]);

                  /* New */
                  printHW(xcomponent);

                  // End New

                  System.exit(0);
            } catch (Exception exception) {
                  System.err.println(exception);
            }
      }

      public static void printHW(XComponent xSc) {

            // getting the text document object
```

```
            XTextDocument xtextdocument = (XTextDocument) UnoRuntime
                        .queryInterface(XTextDocument.class, xSc);
            XText xText = xtextdocument.getText();
            XTextRange xTextRange = xText.getEnd();
            xTextRange.setString("Ouverture d'un document texte.
\n");
            xTextRange = xText.getEnd();
            xTextRange.setString("\nBon, on y va :\n\n");
            xTextRange = xText.getEnd();
            xTextRange.setString("Hello World !\n");

      }// printHW

}
```

C'est une modification de l'exemple DocumentLoader.java livré avec le SDK. Il a été modifié de manière à pouvoir être lancé sur une machine distante. Que fait ce programme ?

① com.sun.star.comp.helper.Bootstrap est une classe qui procure un contexte, une passerelle, un gestionnaire de service, une connexion, enfin tout ce qui permet le « démarrage » d'une application. Sur cette ligne, on obtient le contexte initial.

② On va chercher le gestionnaire de service attaché à ce contexte.

③ Pour obtenir une instance d'une implémentation d' UnoUrlResolver, le service d'accès distant.

④ Ce service ne dispose que d'une interface : XunoUrlResolver qui est associée ici.

⑤ Ce qui fournit une référence de l'application distante (un processus OpenOffice.org qui était déjà lancé ou que notre requête vient de lancer). Sous Linux, il est nécessaire qu'un processus OOo soit déjà lancé. Ce dernier peut ne pas avoir ouvert de document, ni même lancé de « Desktop ».

⑥ Auquel est associé son gestionnaire de composant.

⑦ Auquel on demande une référence à l'interface XpropertySet (le mécanisme qui permet l'association de données manipulées par leur nom).

⑧ On récupère par cette technique le contexte par défaut de l'application distante.

⑨ On lui ajoute un contexte de composant.

⑩ On associe le service com.sun.star.frame.Desktop (Cadre racine de tous les autres) au contexte de composant précédemment obtenu auquel on ajoute

⑪ Une référence à une implémentation de l'interface autorisant le chargement d'un document.

⑫ Ce qui est fait ici où l'on ouvre un nouveau document à l'aide du traitement de

texte.

La suite n'est que la saisie de quelques lignes de texte.

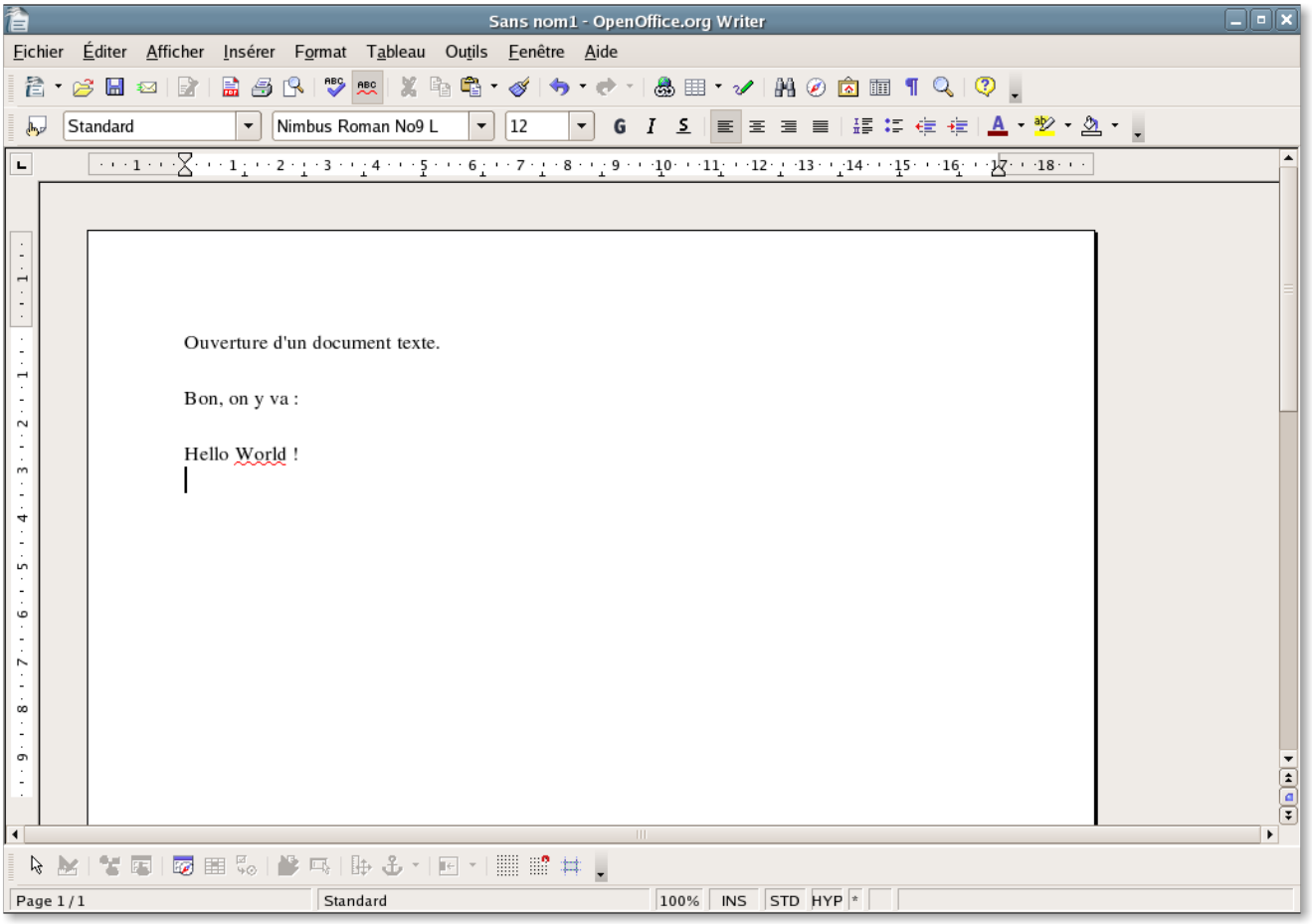
Le détail de ce que font les lignes n'est pas assez explicite ? Essayons une autre explication...

Dans un premier temps, on cherche à obtenir d'une machine distante un objet UNO. Ce qui est fait à la ligne ⑤. Pour cela, il nous faut une instance du service « com.sun.star.bridge.UnoResolver » ③ qui implémente l'interface XunoUrlResolver ④ dont la méthode resolve ⑤ fournit l'objet recherché. Le service « com.sun.star.bridge.UnoResolver » est obtenu du gestionnaire de service ② depuis le contexte initial mis à disposition par le Java Uno Runtime ①.

Une fois obtenu, la référence à cet objet distant, on veut ouvrir le composant « swriter » (traitement de texte) sur cette machine cible. Ce chargement est effectué par la méthode « loadComponentFromUrl » ⑫. Cette méthode est définie par l'interface XcomponentLoader ⑪ mise à disposition par le service « com.sun.star.frame.Desktop » ⑩. Ce service est obtenu en transformant en « composant » l'objet Uno reçu de la machine distante. (Pour raccourcir des phrases déjà longues, je dis « objet » pour « référence à l'objet distant »). Un « composant étant un objet Uno qui dispose d'un gestionnaire de service et d'un contexte, on ajoute le premier en ⑥ et le contexte par défaut à l'aide des lignes ⑦, ⑧ et ⑨. Le gestionnaire de service du composant ainsi fabriqué fournit le service recherché (Desktop) ⑩.

Bien sûr, 10 lignes pour ouvrir le traitement de texte sur une machine distante, c'est un peu long... D'autant que cette procédure est utilisée y compris pour ouvrir une application OOo sur la machine locale. La méthode com.sun.star.comp.helper.Bootstrap.bootstrap() introduite récemment et qui résume les dix premières lignes du code ne semblant pas fonctionner en l'état (telle que distribuée avec le SDK) sur la FC4.

En lançant ce programme, on obtient la fenêtre suivante :



Paramétrage OpenOffice.org

En ⑤, on demande un objet à l'installation distante en envoyant un message à l'aide du mécanisme des « sockets » sur le port 8100. Ce port est celui de nombreux exemples du SDK. Vous ne le trouverez pas associé à OpenOffice.org en http://iana.org/assagnments/port-numbers . Vous pouvez en choisir un autre. Encore faut-il que l'application distante écoute sur le port que vous avez choisi.

Deux méthodes pour faire ça :

1. Vous lancer OpenOffice.org avec la commande suivante :

   soffice '-accept=socket, host=0, port=8100;urp'

2. Ou bien vous modifiez la configuration au lancement d'OpenOfficee en modifiant le fichier :

   /usr/lib/openoffice.org2.0/share/registry/data/org/openoffice/Setup.xcu

   en remplaçant l'étiquette :

   <prop oor:name= « ooSetUpConnectionURL »/>

   par :

```
<prop oor:name="ooSetupConnectionURL">
<value>socket,host=localhost,port=8100;urp;StarOffice.ServiceManager
</value>
</prop>
```

Au prochain lancement, sur cette machine, OpenOffice.org écoutera le port 8100. Il est possible de s'en assurer à l'aide de la commande :

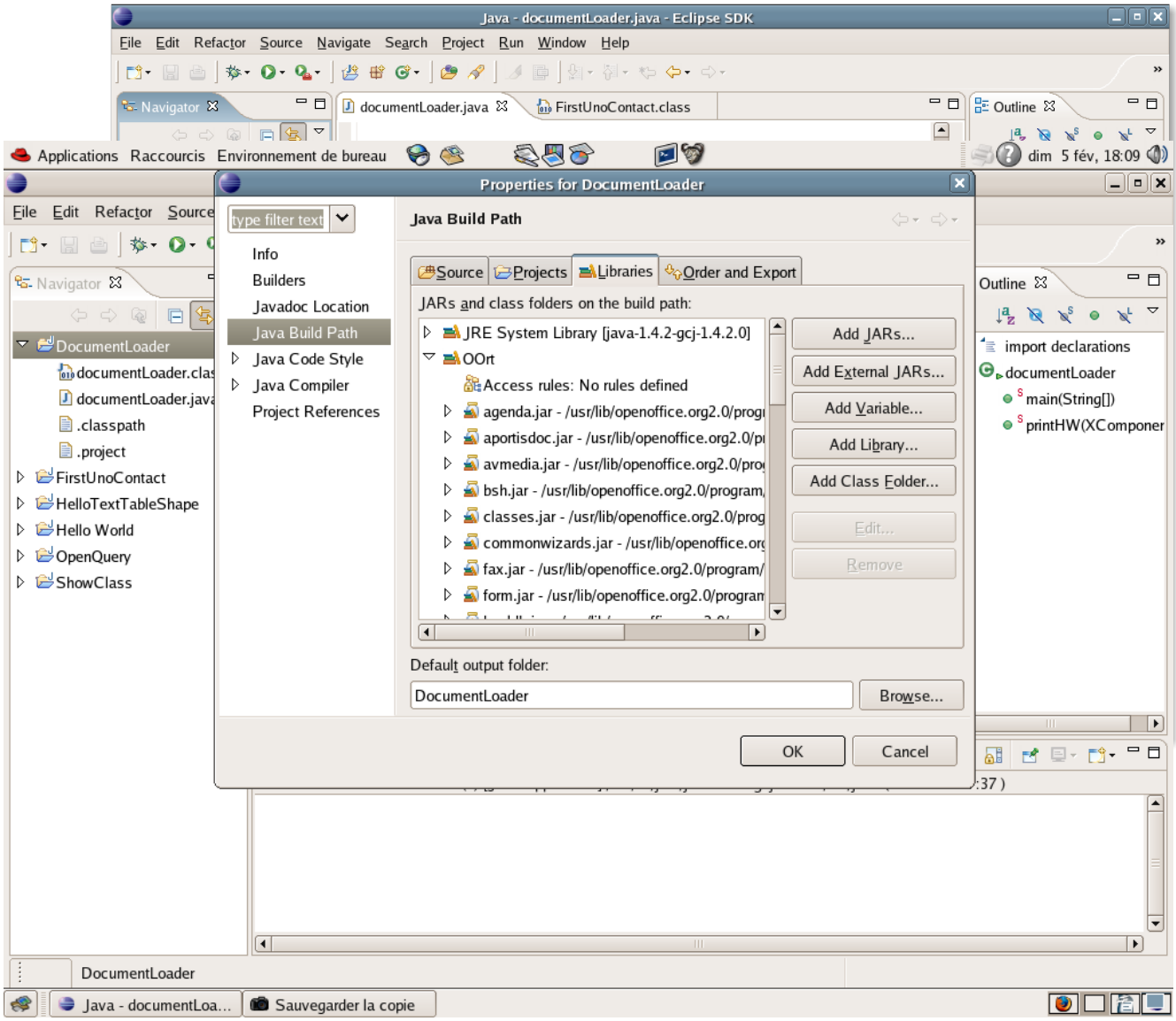
```
netstat -a
```

qui doit lister un processus écoutant sur le port 8100.

Utiliser l'interface de développement Eclipse

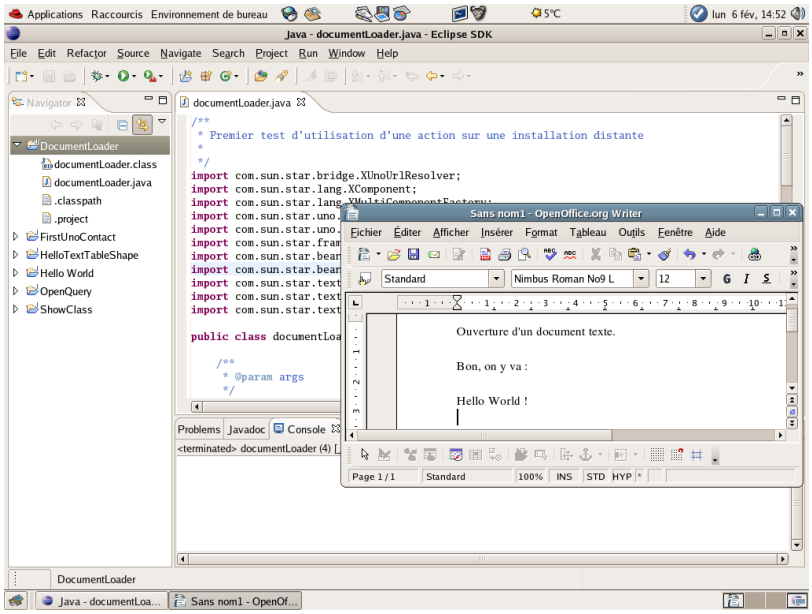
L'environnement de développement Open Source Eclipse http://www.eclipse.org/ compilé avec gcj http://gcc.gnu.org/java/ est disponible dans les paquetages de plusieurs distributions (Debian, Ubuntu, Red Hat, Fedora Core 4).

Il suffit de lancer Eclipse, de créer un nouveau projet nommé par exemple « DocumentLoader » et d'y intégrer le fichier Java listé précédemment. On obtient :



Toutes ces classes ne sont pas nécessaires mais pour un premier test, nous gagnons du temps à ne pas faire de tri.

En lançant l'exécution sous forme de « Java Application » , nous lançons

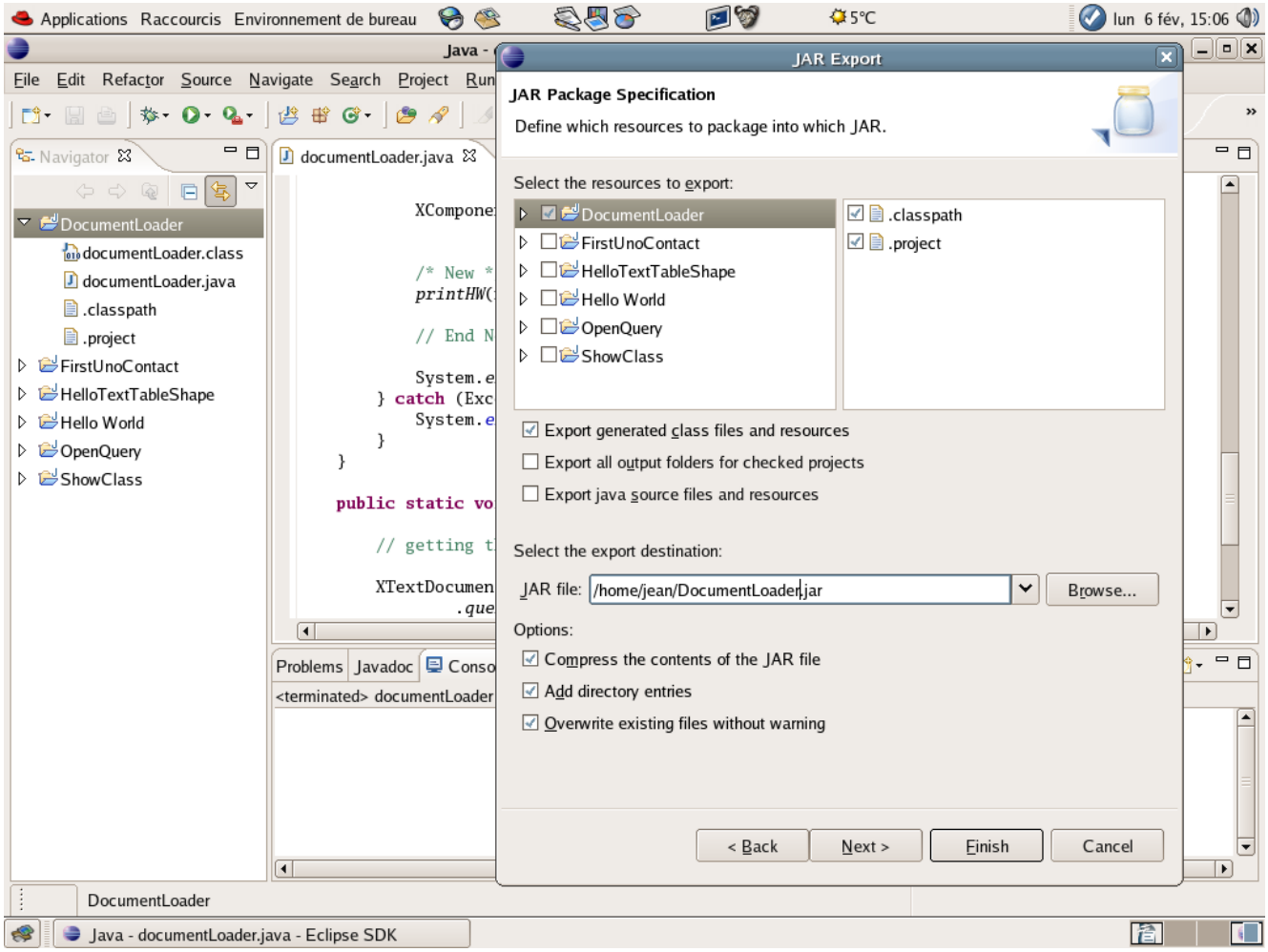


correctement le programme.

Il reste à faire en sorte de pouvoir utiliser ce programme en dehors de l'interface de développement. Il faut donc construire un paquetage le contenant et écrire un script de lancement.

Construction du paquetage

Nous le réalisons sous Eclipse par la commande Fichier>Export> Select (JAR File) :



Nous obtenons un .jar qui est un fichier d'archive compressé. Nous pouvons ouvrir cette archive avec n'importe quel gestionnaire d'archives (File Roller, 7-zip ...). Un fois ouvert, nous lui ajoutons les fichiers nécessaires au chargement de l'UNO java Run Time qui se trouvent dans le répertoire « classes » du SDK, ce sont les répertoires « com » et « win ». Nous ouvrons aussi le répertoire META-INF pour aller modifier le fichier MANIFEST.MF que nous avons généré incorrectement à l'aide d'Eclipse. Nous le remplaçons par :

```
Manifest-Version: 1.0
Main-Class: com.sun.star.lib.loader.Loader
Name: com/sun/star/lib/loader/Loader.class
Application-Class: documentLoader
```

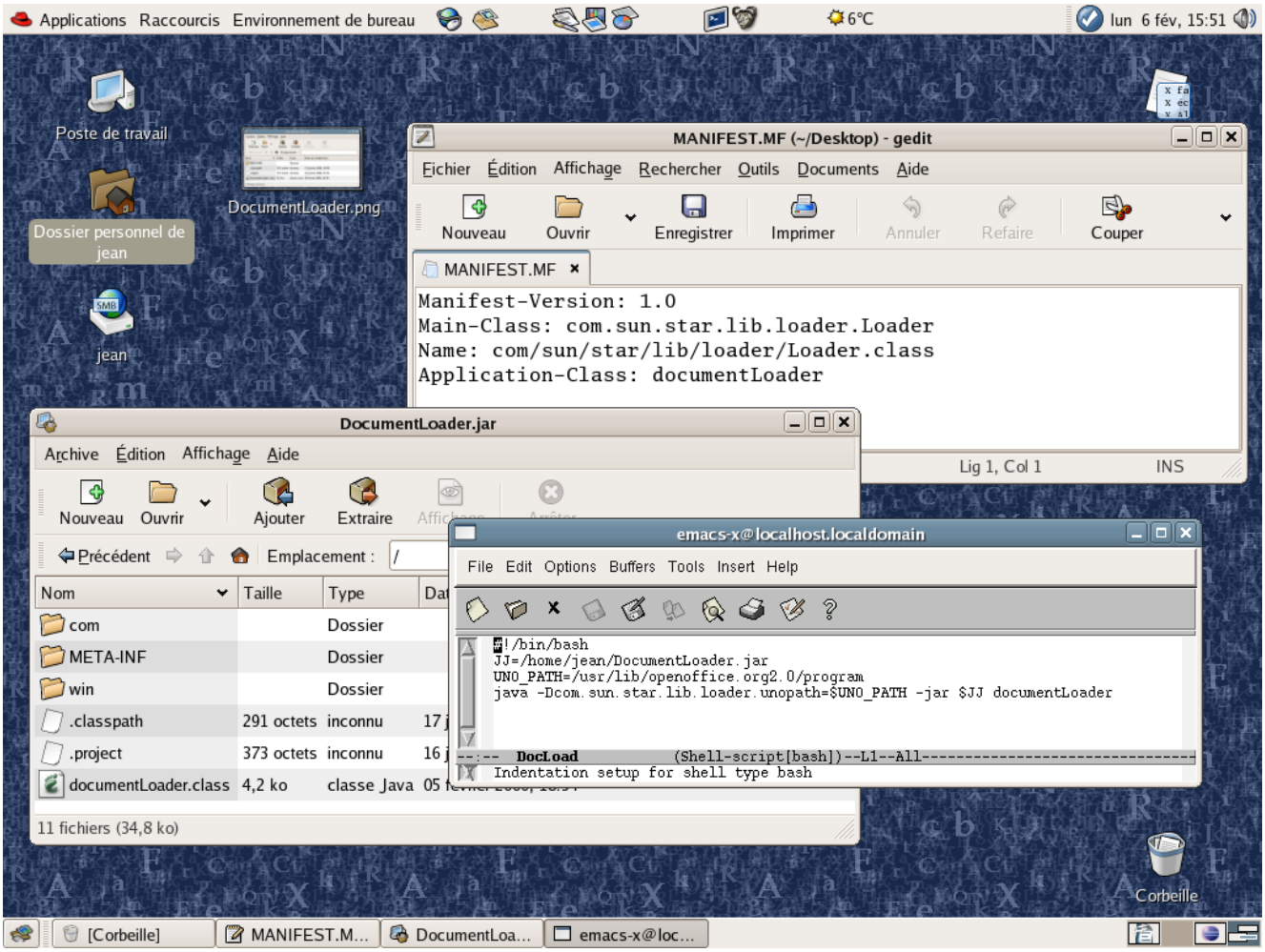
Car il faut utiliser un chargeur spécifique aux objets UNO.

Il reste à construire le script de lancement :

```
#!/bin/bash
JJ=/home/jean/DocumentLoader.jar
UNO_PATH=/usr/lib/openoffice.org2.0/program
```

```
java -Dcom.sun.star.lib.loader.unopath=$UNO_PATH -jar $JJ
documentLoader
```

Nous nommons ce fichier, nous lui donnons les droits d'exécution et nous pouvons le lancer. Nous obtenons l'ouverture d'un nouveau document dans le traitement de texte contenant les quelques lignes demandées.



La figure précédente résume les actions de modification du .jar et de son script de lancement.

Notez que le fichier d'archive obtenu, compilé avec gcj est transférable sur une une autre plateforme (y compris Microsoft!) où il fonctionne de la même manière. Sous Windows, nous n'avons même pas besoin de jouer avec la CLASSPATH, les installateurs s'en sont chargé. Cependant si vous utilisez le Sun JDK 1.5, il vous faut ajouter un fichier INDEX.LIST dans le répertoire META-INF de l'archive. Ce fichier est construit automatiquement par la commande :

```
jar -i DocumentLoader.jar
```

Il suffit ensuite d'exécuter la commande :

```
java -jar DocumentLoader.jar documentLoader
```

Pour lancer le traitement de texte sur la même machine que précédemment.

En guise de conclusion provisoire

Fait assez rare pour un logiciel issu de l'édition commerciale, OpenOffice.org est un logiciel libre (re)bâti sur des concepts à la fois très sophistiqués et très modernes. La complexité sous-jacente peut étonner pour un outil intégré de bureautique sauf si l'on se place dans une perspective de ressources largement distribuées sur un réseau où chacun va chercher sur un ou plusieurs serveurs les « briques » logicielles dont il a besoin pour travailler. Si l'évolution se confirme, nous devrions dans les prochaines versions voir apparaître des « run time » ou clients légers pour les utilisateurs et une évolution de la version actuelle vers un serveur d'application. L'objectif étant pour les organisations de diminuer le coût de la maintenance et des mises à jour. Tout cela est cohérent avec la stratégie d'un constructeur pour lequel « The Network is the Computer » et qui compte bien à l'occasion vendre quelques belles machines et tout ce qui va avec. Rien n'est gratuit dans ce monde, mais jusqu'à présent les choses semblent assez compréhensibles.

Pour vous, si vous voulez développer des applications pour OpenOffice.org : commencez par lire le « Developer's Guide ». Apprenez à vous y retrouver dans la documentation en ligne (api, udk et leurs listes de discussion ...).

Programmez en Java pour comprendre comment ça marche, utilisez le Basic toutes les fois où vous n'êtes pas obligés de faire autrement, étudiez l'interface C++ si vous tenez absolument à voir votre nom dans les « credits » ou que votre application nécessite des calculs longs et répétitifs...

A vos claviers...