Autodocumentation inetdoc

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Le projet inetdoc.LiNUX ne contient pas de code de logiciel ; juste des documents, des images et les Makefiles de traitement. Voici quelques indications sur l'installation d'une copie du projet. Il s'agit juste de générer les pages web et les versions imprimables des documents.

Table des matières

1. Copyright et Licence

1.1. Méta-information

Cet article est écrit avec DocBook XML sur un système Debian GNU/Linux. Il est disponible en version imprimable aux formats PDF : `autodoc.pdf`.

2. Installation rapide

L'installation à partir de sources génère les pages web et les versions imprimables des documents du projet inetdoc.LiNUX.

Pour consulter les pages web vous aurez besoin d'un serveur web configuré pour permettre l'utilisation des pages personnelles. Le répertoire cible par défaut est : `~/public_html/prj/inetdoc`.

Les versions imprimables des documents sont elles aussi placées dans l'arborescence de page web personnelle. Le répertoire cible est : `~/public_html/prj/inetdoc/telechargement/`.

Voici la procédure à suivre :

```
$ wget http://www.linux-france.org/prj/inetdoc/telechargement/inetdoc.src.tar.bz2
$ tar xvjf inetdoc.src.tar.bz2
$ cd lfo/prj/inetdoc
$ make ; make
```

Et voilà, après quelques instants, vous disposez d'une copie locale du projet sur votre disque.

Comment ? L'exécution de la commande make échoue avec un message d'erreur. Y aurait-il un problème de dépendance ? Une application absente ? Vous n'avez qu'à consulter les sections suivantes ;-)).

3. Arborescence d'installation

L'installation par défaut utilise la page personnelle de l'utilisateur actif comme racine : `~/public_html/prj/inetdoc`.

Voir la variable `HTML_WRK` dans le fichier `Makefile.Rules` pour modifier ce répertoire par défaut.

```
.
|-- articles
|   |-- adressage.ipv4
|   |-- devmgmt
|   |-- ethernet
|   |-- modelisation
|   |-- reseau.libre
|   |-- rnis
|   |-- segmentation.lan
|   |-- stockage
|   `-- transmission.tcpip
|-- cours
|   |-- admin.reseau.dns
|   |-- admin.reseau.dns-dhcp
|   |-- admin.reseau.fs
|   |-- admin.reseau.neigh
|   |-- admin.reseau.nfs
|   |-- admin.reseau.nis
|   |-- admin.reseau.synthese-dns
|   |-- admin.reseau.synthese-nfs3-nis
|   |-- admin.reseau.tp
|   |-- archi.tp
|   |-- config.interface
|   |-- explore.gnulinux
|   |-- interco.access-control.tp
|   |-- interco.noyau
|   |-- interco.noyau.synthese
|   |-- interco.noyau.tp
|   |-- interco.ppp.tp
|   |-- interco.rawip.tp
|   |-- interconnexion.fonctions.noyau
|   |-- interconnexion.perimetres
|   |-- intro.analyse
|   |-- routage.inter-vlan
|   |-- services_internet
|   `-- socket.udp.cpp
|-- formations
|   |-- explore.gnulinux
|   |-- fc.cnes
|   |-- m1-stri
|   |-- m2-stri
|   |-- polycop
|   `-- pourquoi.gnulinux
|-- guides
|   |-- Advanced-routing-Howto -> lartc
|   |-- NAT-HOWTO
|   |-- iptables-tutorial
|   |-- lartc
|   |-- netfilter-hacking-HOWTO
|   |-- networking-concepts-HOWTO
|   |-- packet-filtering-HOWTO
|   |-- rnis
|   |-- zebra.ospf
|   |-- zebra.rip
|   `-- zebra.statique
|-- images
|-- liens
|-- notice.legale
|-- securite
|   |-- amavisd-new
|   `-- tutoriel
|-- sources
|   `-- files
`-- telechargement
    `-- xml
```

4. Applications utilisées

Les `Makefile`s de l'arborescence servent à générer les pages web et les versions imprimables des documents. Ces documents utilisent plusieurs formats sources différents. Chaque format source nécessite une chaîne de développement spécifique.

Les fichiers sources des documents du projet utilisent les formats : HTML, DocBook XML, LinuxDoc, MagicPoint et OpenOffice Impress. Voici, pour chaque type de source, la liste des paquets/applications associées. Les noms de paquets sont ceux de la distribution Debian.

Pour installer le tout :

```
# apt-get install perl sed tidy xsltproc fop \
> libxml2-utils docbook docbook-xml docbook-xsl docbook-utils \
> linuxdoc-tools linuxdoc-tools-latex mgp gs xpdf tetex tetex-extra
```

Quelques indications sur les paquets en question :

Documents source de type (X)HTML

- sed : The GNU sed stream editor. Corrections manuelles de balises, (insertion|extraction) de fichiers patrons.
- tidy : outil de validation syntaxique et de mise en forme des fichiers (X)HTML.

Documents source de type DocBook XML

- libxml2-utils : outil xmllint de validation syntaxique, de remise en forme des documents XML et de fusion des entités DocBook XML.
- docbook-xml : système de représentation XML standard pour les documents techniques. Ce système est une «collection» de balises qui permet de structurer un document en chapitres, sections, listes, etc.
- docbook-xsl : feuilles de styles XSL standard pour la transformation des fichiers sources DocBook XML vers les formats (X)HTML et FO.

Documents source de type LinuxDoc

- linuxdoc-tools : conversion SGML pour les documents écrits avec le jeu de définitions LinuxDoc DTD. Ce jeu de définition est utilisé pour la publication de documentations «historiques» telles que les Howtos Netfilter.

Présentations

- MagicPoint ou mgp : outil de présentation de vues/transparents basé sur les bibliothèques graphiques du projet X.Org.
- OpenOffice.org Impress : outil de présentation de vues/transparents de la suite OpenOffice.org.

Formats imprimables PDF et Postscript

- fop ou fo processor : transformation des fichiers sources DocBook XML vers le format PDF.
- xpdf : outils de traitement des fichiers Postscript et PDF.

5. Génération des fichiers PDF avec FOP et Java

Apache FOP (Formatting Objects Processor) est l'outil qui permet de générer les versions imprimables au format PDF à partir des documents sources DocBook XML et d'une feuille de styles XSL-FO. Ce processeur est développé en Java et c'est incontestablement l'outil libre le plus avancé sur la production de versions imprimables.

La version du Java Runtime Environment (JRE) influe beaucoup sur le fonctionnement de fop. Par conséquent, il est préférable d'utiliser une version la plus récente possible. Dans ce document on s'appuie sur les paquets construits à partir de la version libre de la chaîne de développement Java baptisée OpenJDK.

5.1. Paquets Java Debian

Pour une installation Java «standard» comprenant la chaîne de développement, on obtient la liste suivante :

```
$ dpkg -l *openjdk* | grep ^ii
ii  openjdk-6-jdk           6b16-1.6.1-2  OpenJDK Development Kit (JDK)
ii  openjdk-6-jre           6b16-1.6.1-2  OpenJDK Java runtime, using Hotspot JIT
ii  openjdk-6-jre-headless  6b16-1.6.1-2  OpenJDK Java runtime, using Hotspot JIT (headless)
ii  openjdk-6-jre-lib       6b16-1.6.1-2  OpenJDK Java runtime (architecture independent libraries)
```

Comme il est possible de faire coexister plusieurs versions de la chaîne de développement Java sur le même système, on doit s'assurer que la version voulue sera bien utilisée par les autres outils.

Note

Avec cette version libre de Java, le fameux plugin est fourni avec le paquet baptisé `icedtea6-plugin`.

```
## update-java-alternatives -v --jre --plugin -s java-6-openjdk
resetting java alternatives
```

Il reste ensuite à vérifier que le paquet correspondant au processeur fop est bien installé :

```
# dpkg -l fop* | grep ^ii
ii  fop               1:0.95.dfsg-5       XML to PDF Translator
```

5.2. Installation de fop à partir du dépôt SVN

Pour bénéficier des toutes dernières avancées sur le développement de fop, il est nécessaire de construire l'outil à partir de ses autodoc. Pour ce faire, il faut télécharger les sources à partir du dépôt du serveur Subversion du projet. On passe ensuite au opérations de construction avec ant, l'équivalent de make dans l'environnement Java.

La récupération de l'arborescence des sources du projet fop se fait avec svn la commande principale du paquet subversion :

```
:~$ mkdir ~/SVN/fop
:~$ cd ~/SVN/fop/
:~/SVN/fop$ svn co https://svn.apache.org/repos/asf/xmlgraphics/fop/trunk
<snip/>
A    trunk/examples/embedding/README
A    trunk/examples/embedding/build.xml
 U   trunk
 Révision 410929 extraite.
```

Après ce téléchargement initial, toutes les opérations de mises à jour se font à l'aide de la commande `svn update`.

On construit ensuite le processeur à l'aide des directives données à la commande ant. La première instruction `ant clean` élimine toute trace de fichiers binaires issus de la construction précédente et la seconde `ant package` construit les paquets jar nécessaires à l'utilisation de fop.

Nettoyage de l'arborescence de développement.

```
:~/SVN/fop/trunk$ ant clean
Buildfile: build.xml

clean:
   [delete] Deleting directory /home/phil/SVN/fop/trunk/build

   BUILD SUCCESSFUL
   Total time: 0 seconds
```

Compilation de la nouvelle version de fop.

```
:~/SVN/fop/trunk$ ant package
Buildfile: /home/phil/SVN/fop/trunk/build.xml
Trying to override old definition of task javac
Trying to override old definition of task junit

init-avail:
     [echo] ------------------- Apache FOP svn-trunk [1999-2010] ----------------
     [echo] See build.properties and build-local.properties for additional build settings
     [echo] Apache Ant version 1.8.0 compiled on March 11 2010
     [echo] VM: 19.1-b02, Sun Microsystems Inc.
     [echo] JAVA_HOME: ${env.JAVA_HOME}
     [echo] JAI Support NOT Present
     [echo] JCE Support PRESENT
     [echo] JUnit Support PRESENT
     [echo] XMLUnit Support PRESENT

init:
<snip/>

package:

BUILD SUCCESSFUL
Total time: 8 seconds
```

On peut maintenant vérifier que l'outil est disponible.

```
:~/SVN/fop/trunk$ ./fop -version
FOP Version svn-trunk
```

Dernière étape essentielle, il faut positionner la variable `FOP` fichier principal des règles de génération des pages XHTML et des fichiers imprimables : `Makefile.Rules`.

```
# fop -> Génération FO-PDF
FOP=~/SVN/fop/trunk/fop
```

Voilà, c'est prêt !. On dispose maintenant d'un processeur qui va permettre de générer des documents PDF en quantités industrielles. Il ne reste plus qu'à rédiger ;)).

6. Ajout d'un nouveau document

Pour ajouter un nouveau document, il faut créer un répertoire dans lequel on place le fichier source et un `Makefile` pour automatiser les traitements.

6.1. Contenu d'un Makefile

Voici un patron de `Makefile` et quelques indications sur ses variables et fonctions :

`DIR`

Nom du répertoire de travail courant dans lequel on place le(s) fichier(s) source(s) à traiter.

`DEPTH`

Profondeur du répertoire de travail relativement à la racine de l'arborescence de travail.

`SUBDIRS`

Liste des sous-répertoires à traiter. Dans le cas du projet inetdoc.linux, un document DocBook de type «bibliographie» pointe vers plusieurs autres documents. Chacun de ces documents est placé dans un sous-répertoire différent.

`PROCESS_TYPE`

Liste des traitements à effectuer dans le répertoire de travail. Les différents types de traitements sont détaillés dans Section 6.2, « Types de traitements ».

`XML_FILES`

Dans le cas d'un document DocBook ; liste des fichiers source. Dès qu'un document de cette liste est modifié, il faut reprendre le traitement

`SYMLINKS`

Liste des liens symboliques à créer. Un document peut reprendre des éléments contenus dans d'autres répertoires. Les liens symboliques sont alors beaucoup plus utiles que les copies.

`all: dirs subdirs $(PROCESS_TYPE)`

Somme des traitements à effectuer :

- `dirs` : création des répertoires contenus dans la liste `DIR`.
- `subdirs` : création des sous-répertoires contenus dans la liste `SUBDIRS`.
- `$(PROCESS_TYPE)` : traitements spécifiques au(x) document(s) donnés dans la liste `PROCESS_TYPE`.
- autres traitements : dans le cas du répertoire `sources`, il s'agit de générer le fichier de distribution des sources du projet.

6.2. Types de traitements

L'ensemble des traitements existants sont définis dans le fichier de règles : `Makefile.Rules`. C'est ce fichier qu'il faut consulter pour connaître de détail des opérations effectuées.

`dirs`

Création des répertoires dans l'arborescence cible à partir de la racine définie par `HTML_WRK`.

`subdirs`

Liste des sous-répertoires dans lesquels d'autres documents sont à traiter.

`clean`

Nettoyage de tous les fichiers intermédiaires générés lors des traitements. La liste des fichiers à effacer est extraite des propriétés `svn:ignore` des répertoires du dépôt Subversion (SVN).

`symlink`

Création des liens symboliques dans le répertoire de travail.

`file`

Copie simple de fichiers dans l'arborescence cible.

`html`

Passage à la lessiveuse tidy et copie du résultat dans l'arborescence cible.

`ld2www`

Génération des pages web à partir d'une source LinuxDoc sans oublier le passage par la lessiveuse.

`ld2print`

Génération des formats imprimables Postscript et PDF à partir d'une source LinuxDoc.

`_xml2xhtml`

Génération des pages web à partir d'une source DocBook sans oublier le passage par la lessiveuse.

`_xml2print`

Génération des formats imprimables Postscript et PDF à partir d'une source DocBook.

`_mgp2xhtml`

Génération d'une page web d'index des vues d'une présentation MagicPoint ainsi que d'une page web par vue.

`_mgp2print`

Génération des formats imprimables Postscript et PDF à partir d'une présentation MagicPoint.

`_mgp2db`

Construction automatique d'un article DocBook à partir des copies d'écran des vues de présentation MagicPoint.

`odp2xhtml`

Génération des pages web d'index et de vues à partir de l'exportation PDF d'une présentation OpenOffice.org Impress.

7. Journal des mises à jour : Changelog

La documentation se gère comme le code source d'un logiciel. Les fichiers XML et les `Makefiles` représentent de code source du projet. Pour gérer les évolutions d'un volume important de code source dans le temps, il est essentiel d'utiliser un gestionnaire de contrôle de version.

7.1. Génération de la page ChangeLog avec CVS

Depuis le début du projet inetdoc.LiNUX en 2001 jusqu'en Novembre 2005, le gestionnaire de contrôle de version était CVS. Pour générer le journal au format XML, il est nécessaire d'utiliser un script fourni dans le paquet du même nom `cvs2cl`. Une fois le journal créé au format XML, on lui applique une feuille de styles XSL pour produire la page web «lisible» au format XHTML.

Les règles de génération de la page web contenues dans le `Makefile` sont :

La feuille de styles XSL utilisée pour produire la page web se présente comme suit :

7.2. Génération de la page ChangeLog avec SVN

Depuis Novembre 2005, le gestionnaire de contrôle de version des fichiers sources utilisé pour le projet inetdoc.LiNUX est `subversion` (AKA SVN). Ce gestionnaire présente de nombreux avantages relativement au précédent (CVS). Il est possible de générer directement le journal des mises à jour au format XML sans avoir recours à un paquet supplémentaire. Il est cependant toujours nécessaire d'utiliser une feuille de styles XSL pour produire la page web au format XHTML.

Depuis Avril 2007, le journal des mises à jour au format XML est modifié à l'aide d'un script perl de façon à ajouter les statistiques produites par la commande diffstat pour chaque fichier modifié. L'objectif de cette modification est de fournir une information sur l'importance des éditions sur les fichiers autodoc.

Les règles de génération de la page web contenues dans le `Makefile` de génération et de publication du journal des éditions de fichiers source (Changelog) sont :

```
.PHONY: $(TARGET_DIR)/README.html
$(TARGET_DIR)/README.html: common/asvn2cl.xsl
     svn log --xml --verbose \
     -r{`date --date 'tomorrow' +%Y-%m-%d`}:{`date --date '9 months ago' +%Y-%m-%d`} \
     >changes.xml
     ./common/extract_changelog_diffstat.pl <changes.xml >changes.xml.new
     xsltproc -o $@ common/asvn2cl.xsl changes.xml.new
     rm -f changes.xml*
```

Le script Perl de modification du journal XML effectue les transformations de balises suivantes :

- Résultat produit par la commande `svn log --xml`.

```
<paths>
  <path
    action="M">/trunk/prj/inetdoc/sources/autodoc.xml<path>
<paths>
```

- Résultat après transformation par le script `extract_changelog_diffstat.pl`.

```
<objects>
<object>
<path
  action="M">/trunk/prj/inetdoc/sources/autodoc.xml<path>
  <stats>14 ++++++++++++++<stats>
  <object>
<objects>
```

Le script Perl `extract_changelog_diffstat.pl` de modification du journal est donné dans l'Annexe A, Script de transformation du journal XML de Subversion.

A. Script de transformation du journal XML de Subversion

Ce script Perl modifie les balises XML du journal produit par la commande `svn log --xml` en y ajoutant le résultat de la commande diffstat. Le nouveau jeu de balises est donné dans la Section 7.2, « Génération de la page ChangeLog avec SVN ».