Guide pratique de démystification de DocBook

Version française du guide pratique DocBook Demystification HOWTO

Eric Raymond

<esr CHEZ thyrsus POINT com>

Adaptation française: Raphaël Semeteys

Relecture de la version française: Jean-Philippe Guérard

Préparation de la publication de la v.f.: Jean-Philippe Guérard

Version : 1.3.fr.1.0

7 janvier 2006

Historique des versions		
Version 1.3.fr.1.0	2006-01-07	RSS, JPG
Adaptation française initiale.		
Version 1.3	2004-02-27	ESR
Ajout de pointeurs vers deux éditeurs. *Add pointers to two editors.*		
Version 1.2	2003-02-17	ESR
Déplacement des références à SGML après que celui-ci ait été présenté. *Reorder to defer references to SGML until after it has been introduced.*		
Version 1.1	2002-10-01	ESR
Correction d'une erreur faite par inadvertance sur la position de la FSF. Ajout d'un pointeur vers la FAQ DocBook. *Correct inadvertent misrepresentation of FSF's position. Added pointer to the DocBook FAQ.*		
Version 1.0	2002-09-20	ESR
Version initiale. *Initial version.*		

Résumé

Ce guide pratique tente de dissiper le brouillard et le mystère entourant le système de balisage DocBook et les outils associés. Il est destiné aux auteurs de documentations techniques pour les projets libres sous Linux, mais devrait également être utile à la rédaction d'autres types de documents sur d'autres systèmes Unix.

Table des matières

1. Introduction

Bon nombre de grands projets libres sont en train de converger vers l'utilisation du format DocBook comme format standard de documentation — des projets comme le noyau Linux, GNOME, KDE, Samba et le Projet de documentation Linux (LDP). Les défenseurs du balisage structurel basé sur XML (par opposition au style plus ancien de balisage de présentation illustré par troff, Tex et Texinfo) semblent avoir gagné la bataille théorique. Il est possible de produire un balisage de présentation à partir d'un balisage structurel, mais l'inverse est très difficile.

Néanmoins, une grande confusion entoure DocBook et les programmes qui l'accompagnent. Ses adeptes parlent un argot dense et intimidant, même selon les canons de l'informatique, utilisant à tout va des sigles sans rapport évident avec ce qui doit être fait pour écrire du texte balisé et produire du HTML ou du Postscript. Les standards et les documents techniques sur XML sont notoirement obscurs.

Ce guide va tenter de clarifier les principaux mystères existants autour de DocBook et de son utilisation pour les documents relatifs aux logiciels libres — qu'il s'agisse de documentations techniques ou politiques. Notre objectif est de vous permettre de comprendre non seulement ce que vous devez faire pour fabriquer des documents, mais aussi pourquoi le processus est si complexe — et comment on peut s'attendre à voir cela changer au fur et mesure de la disponibilité de nouveaux outils DocBook.

2. Pourquoi s'intéresser à DocBook ?

DocBook offre deux possibilités qui le rendent vraiment intéressant. La première est la *production multi-formats* et la seconde la réalisation de *bases de documents interrogeables*.

La *production multi-formats* est la possibilité la plus simple et la plus proche de la réalisation ; il s'agit de la capacité à écrire un document dans un format maître unique, qui puisse être utilisé pour produire différents formats d'affichage (en particulier du HTML pour la visualisation en ligne et du Postscript pour des impressions de haute qualité). Cette capacité est désormais assez bien mise en œuvre.

Le terme *base de documents interrogeable* est une manière condensée d'expliquer que DocBook puisse nous aider à aller vers un monde où l'ensemble de la documentation présente sur votre système d'exploitation libre constituera une base de données hypertexte, indexée et interrogeable de textes enrichis (plutôt que d'être éparpillée dans des formats variés en de multiples endroits comme c'est le cas aujourd'hui).

Idéalement, lorsque vous installerez un logiciel sur votre machine, il enregistrera sa documentation DocBook dans le catalogue de votre système. Celui-ci produira automatiquement du HTML correctement indexé et interconnecté au HTML contenu dans le reste de votre catalogue. La documentation du nouveau paquet sera alors disponible via un navigateur. Des recherches pourront être réalisés sur l'ensemble de la documentation via une interface ressemblant à un bon moteur de recherche.

Le HTML en tant que tel n'est pas un format suffisamment riche pour nous rapprocher de ce monde. Pour ne citer qu'une seule de ses lacunes, le HTML ne permet pas de déclarer explicitement des entrées d'index. DocBook *possède* la richesse sémantique permettant la réalisation de bases de documents structurées. C'est la raison fondamentale de son adoption par de si nombreux projets.

DocBook a les inconvénients de ses avantages. Certains le trouvent trop lourd et verbeux pour pouvoir réellement constituer un format de composition confortable. Ce qui ne posera pas de problème, tant que les formats qu'ils affectionnent (comme le format POD de Perl ou le Texinfo GNU) disposent de moteurs permettant de produire du DocBook, tout le monde sera satisfait. Peu importe que tous le monde écrive au format DocBook au pas. Du moment que DocBook devient le format commun d'échange de documents, nous resterons capables de réaliser des bases de documents interrogeables unifiées.

3. Balisage structurel : préambule

Les anciens langages de mise en forme tels que Tex, Texinfo ou Troff permettaient de réaliser un *balisage de présentation*. Les instructions que vous donniez à ces systèmes concernaient l'apparence et la disposition physique du texte (par exemple, des modifications de la fonte ou de l'indentation).

Tant que votre objectif restait d'imprimer sur un seul support ou un seul type de périphérique d'affichage, le balisage de présentation restait une bonne solution. Vous en atteignez les limites lorsque vous commencez à baliser un document en voulant (a) qu'il puisse être mis en forme pour des dispositifs d'affichages très différents (comme des imprimantes et le web) ou (b) réaliser des recherches sur le document et l'indexer via sa structure logique (par exemple pour l'incorporer dans un système hypertexte).

Pour disposer de ces capacités, vous aurez besoin d'un système de *balisage structurel*. Avec le balisage structurel, vous ne décrirez pas l'apparence physique du document mais plutôt les propriétés logiques de ses différentes parties.

Par exemple : dans un langage de balisage de présentation, si vous désirez accentuer un mot, vous indiquerez à l'outil de mise en forme de le mettre en caractères gras. Dans troff(1) cela ressemblera à ceci[1 [p 14]] :

```
Toutes vos base
.B sont
appartiennent à nous !
```

Dans un langage de balisage structurel, vous indiquerez à l'outil de mise en forme de mettre le en relief :

```
Toutes vos base <emphasis>sont</emphasis>
appartiennent à nous !
```

Le « `<emphasis>` » et le « `</emphasis>` » de la ligne précédente sont appelés des *balises de marquage* ou tout simplement des *balises*. Elles constituent les instructions de votre outil de mise en forme.

Dans un langage de balisage structurel, l'apparence physique du document final sera contrôlée par l'application d'une *feuille de style*. C'est la feuille de style qui indiquera à l'outil de mise en forme de « représenter la mise en relief sous forme de caractères gras ». Un des avantages des langages de balisage structurel est qu'en modifiant une feuille de style vous pourrez modifier de manière globale la présentation du document (pour utiliser des polices de caractères différentes par exemple) sans avoir à modifier chaque occurrence, par exemple, de .B dans le document lui-même.

4. Définitions de types de documents (DTD)

Note

Pour rester simple dans notre explication, cette section comporte certains raccourcis historiques qui seront corrigés dans une section ultérieure [p 11] .

DocBook est un langage de balisage structurel. Plus spécifiquement c'est un dialecte XML. Un document DocBook est un fichier XML qui utilise des balises XML pour définir sa structure.

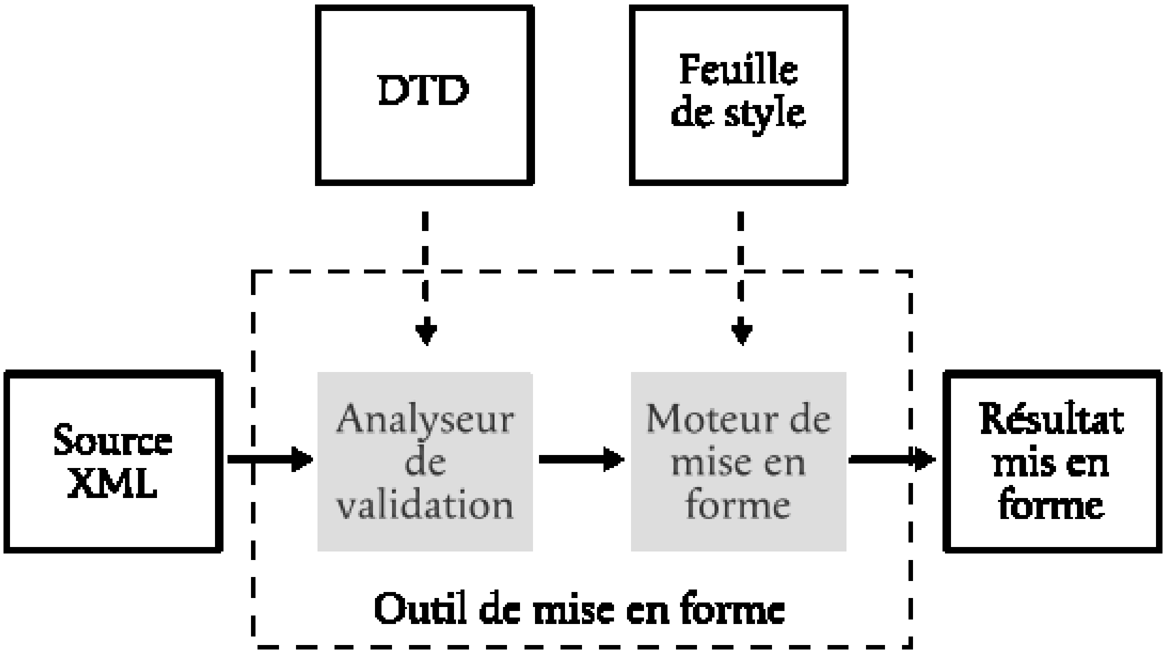
Afin d'appliquer une feuille de style à votre document et de lui donner une belle apparence, un outil de mise en forme aura besoin de savoir certaines choses sur la structure complète du document. Par exemple, il aura besoin de savoir qu'un livre est normalement constitué de parties liminaires, d'une suite de chapitres, puis de parties annexes afin de pouvoir mettre en forme correctement les en-têtes de chapitres. Afin qu'il puisse le savoir, vous devrez lui fournir une *définition de type de document* ou DTD. La DTD indique à l'outil de mise en forme les éléments qui peuvent se trouver dans la structure du document ainsi que l'ordre dans lequel ils peuvent apparaître.

Lorsqu'on décrit DocBook comme une application de XML, on veut dire par là qu'il s'agit d'une DTD — une DTD plutôt conséquente avec près de 400 balises.

Derrière DocBook se cache un type de programme appelé un *analyseur de validation*. Lorsque vous mettez en forme un document DocBook, la première étape à franchir est de le passer au crible d'un analyseur de validation (qui est le premier composant de l'outil de mise en forme DocBook). Ce programme vérifie la validité de votre document par rapport à la DTD DocBook. Cela permet de s'assurer que vous n'êtes en conflit avec aucune des règles structurelles de la DTD (sinon le composant de l'outil de mise en forme en charge de l'application de la feuille de style pourrait s'y perdre).

L'analyseur de validation va soit vous afficher des messages d'erreurs relatifs aux endroits où la structure du document est incorrecte, soit traduire le document en un flux d'*évènements de mise en forme* qui sera finalement combiné avec votre feuille de style pour produire le résultat mis en forme.

Voici un schéma du processus complet :



La partie du schéma comprise dans la zone en pointillés est votre outil de mise en forme, autrement appelée votre *chaîne logicielle* de mise en forme. Pour comprendre ce qui suit, en plus de l'entrée évidente et visible de l'outil de mise en forme (le document source) vous devrez garder en tête les deux autres entrées « cachées » (DTD et feuille de style) de l'outil de mise en forme.

5. Autres DTD

Un court détour par les autres DTD vous aidera à distinguer les parties de la section précédente qui sont spécifiques à DocBook de celles concernant tous les langages de balisage structurel.

TEI (Text Encoding Initiative) est une DTD conséquente et élaborée, principalement utilisée dans les milieux académiques pour la transcription informatique de textes littéraires. La chaîne logicielle de TEI, fonctionnant sous Unix, utilise nombre d'outils utilisés par DocBook, mais avec des feuilles de style et (évidemment) une DTD différentes.

XHTML, la dernière version de HTML, est également une utilisation d'XML décrite par une DTD, ce qui explique l'air de famille existant entre les balises XHTML et DocBook. La chaîne logicielle XHTML est constituée de navigateurs web et d'un certain nombre d'utilitaires spécifiques d'impression.

De nombreuses autres DTD XML sont maintenues afin de faciliter les échanges d'informations structurées dans des domaines aussi divers que l'informatique biologique ou la banque. Vous pouvez consulter la liste des référentiels pour vous faire une idée de la variété des DTD existantes.

6. La chaîne logicielle DocBook

Le moyen le plus simple de mettre en forme et de produire des documents XML DocBook est d'utiliser la chaîne logicielle xmlto. Elle est intégrée à la distribution Red Hat et les utilisateurs de Debian peuvent la récupérer via la commande **`apt-get install xmlto`**.

Pour produire du XHTML à partir de vos sources DocBook, il vous faudra normalement faire quelque-chose comme ceci :

```
bash$ xmlto xhtml foo.xml
bash$ ls *.html
ar01s02.html ar01s03.html ar01s04.html index.html
```

Dans cet exemple, vous avez converti un document XML Docbook nommé `foo.xml` composé de trois sections principales en une page d'index et trois parties. Produire une seule page est tout aussi simple :

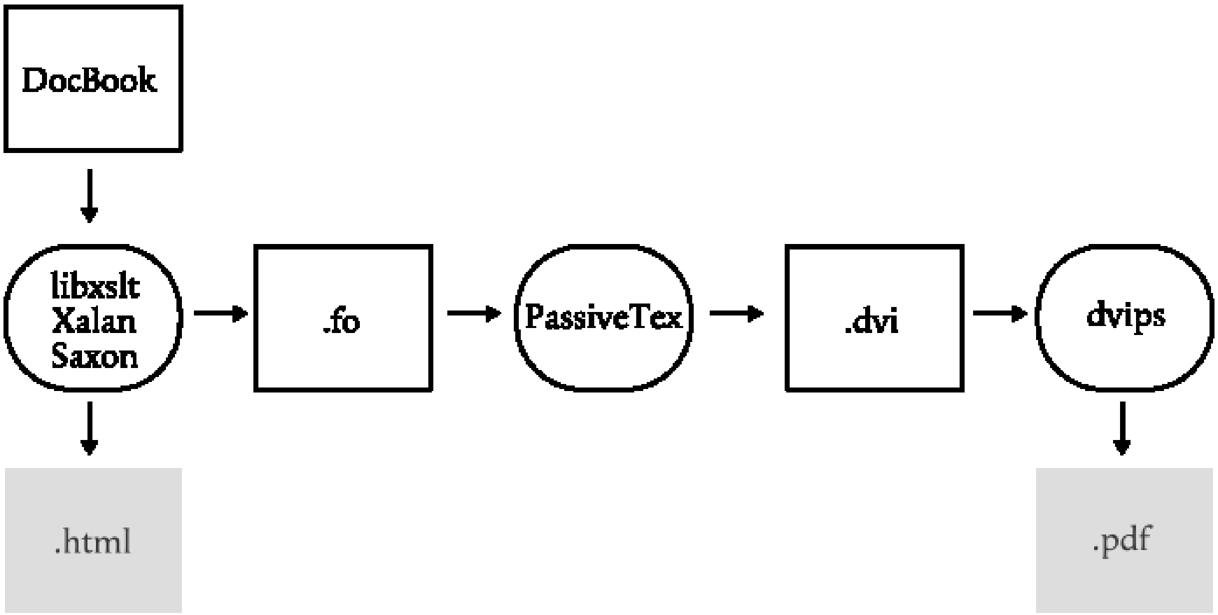
```
bash$ xmlto xhtml-nochunks foo.xml
bash$ ls *.html
foo.html
```

Enfin, voici comment produire du PostScript pour l'impression :

```
bash$ xmlto ps foo.xml     # Production de PostScript
bash$ ls *.ps
foo.ps
```

Certaines versions plus anciennes de **xmlto** peuvent être plus bavardes en émettant de messages du type « Conversion en XHTML en cours » et ainsi de suite.

Pour convertir vos documents en HTML ou en PostScript, vous devez disposer d'un moteur capable d'appliquer à la fois la DTD DocBook et une feuille de style adéquate à votre document. Voici comment s'intègrent ensemble les outils libres utilisés à cette fin :



Chaîne logicielle XML DocBook actuelle

L'analyse du votre document et l'application de la feuille de style vont être réalisées par l'un des programmes suivants. Le plus courant est xsltproc, l'analyseur inclus dans la distribution Red Hat depuis la version 7.3. Les autres possibilités sont les programmes Java Saxon et Xalan.

XHTML étant simplement une autre DTD XML, il est relativement aisé de produire du XHTML de haute qualité à partir de DocBook. La conversion en HTML est réalisée en appliquant une feuille de style plutôt simple et c'est à peu près tout. De même RTF est simple à produire de cette manière et il est facile de produire un fichier de texte brut à partir d'XHTML ou de RTF.

Le cas de l'impression est moins évident. Il est difficile de produire des impressions de haute qualité (ce qui correspond en pratique au format PDF[2 [p 14]] d'Adobe, une version prête à l'emploi de Post-Script). Pour le faire correctement, il est nécessaire de reproduire de manière algorithmique la finesse du jugement humain utilisé par un typographe lorsqu'il passe du niveau du contenu à celui de la présentation.

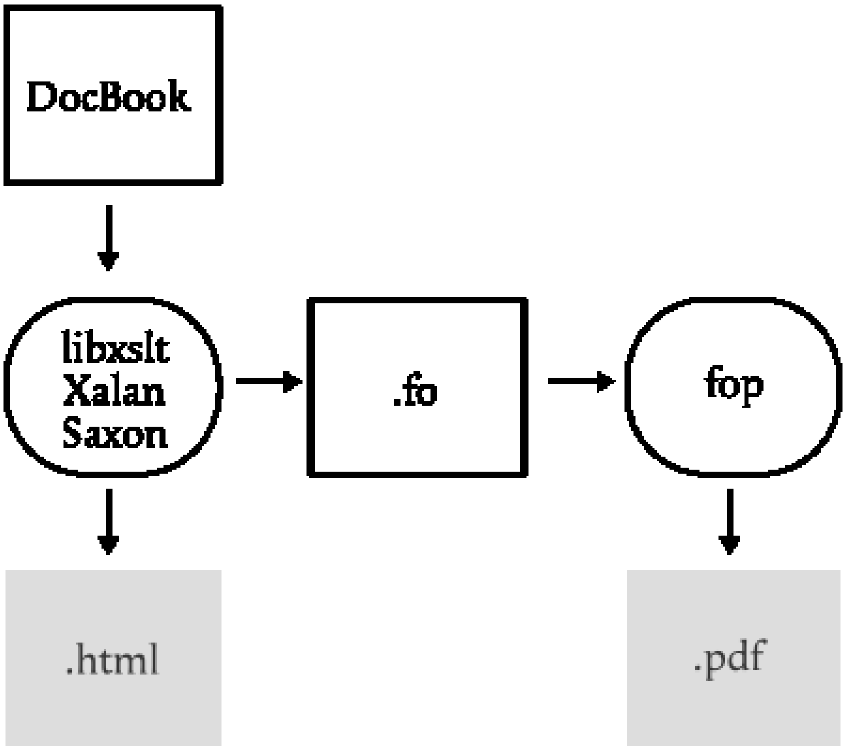
Donc, premièrement, une feuille de style traduit le balisage structurel DocBook en un autre langage XML — FO (*Formatting Objects*). Le balisage FO est un vrai balisage de présentation. Vous pouvez le considérer comme un équivalent fonctionnel de troff dans le monde XML. Il doit être traduit en Postscript pour pouvoir être intégré à un PDF.

Dans la chaîne logicielle intégrée à Red Hat, ce travail est pris en charge par un paquet de macros TeX appelé PassiveTeX. Il traduit dans le langage TeX de Donald Knuth les objets de mise en formes produits par **xsltproc**. TeX a été un des tous premiers projets libres, un langage de mise en forme (au niveau de la présentation) ancien mais puissant, apprécié des mathématiciens (à qui il fournit des fonctionnalités particulièrement évoluées de description des notations mathématiques). TeX est également connu pour son efficacité dans les tâches typographiques de base comme le crénage, le remplissage de lignes ou la césure. Le résultat de l'exécution de TeX, récupéré dans un format appelé DVI (*DeVice Independent*), est ensuite transformé en PDF.

Si vous trouvez que cet enchaînement de XML vers des macros TeX vers DVI vers PDF ressemble à une usine à gaz, vous n'avez pas tort. Ça grince, ça siffle et ça comporte d'horribles verrues. Les fontes constituent un problème important du fait qu'XML, TeX et PDF possèdent des modèles très différents d'utilisation des fontes. En outre, la gestion de l'internationalisation et des paramètres régionaux est un vrai cauchemar. La seule chose qui justifie cet enchaînement est que cela fonctionne.

La solution élégante sera FOP, un traducteur FO vers PostScript développé par le projet Apache. Avec FOP, le problème de l'internationalisation est, s'il n'est pas résolu, bien circonscrit : les outils XML manient l'Unicode de bout en bout. La correspondance entre glyphes et fonte est également un problème relevant strictement de FOP. Le seul problème de cette approche est que cela ne fonctionne pas — du moins pour le moment. En date d'août 2002, FOP est en version alpha non finalisée — utilisable mais brut de décoffrage et fonctionnellement incomplet.

Voici à quoi ressemble la chaîne logicielle FOP :



Future chaîne logicielle XML DocBook intégrant FOP

FOP a des concurrents. Il existe un autre projet appelé xsl-fo-proc qui vise le même objectif que FOP, mais en C++ (et donc plus rapide et ne reposant pas sur l'environnement Java). En date d'août 2002, xsl-fo-proc est aussi en version alpha non finalisée, dans un état assez proche de celui de FOP.

7. Quels sont les projets et les acteurs ?

La DTD DocBook elle-même est maintenue par le Comité technique DocBook (*DocBook Technical Committee*), dirigé par Norman Walsh. Norm est le principal auteur des feuilles de style DocBook. Il a consacré depuis de nombreuses années une grande quantité d'énergie et de talent aux problèmes extrêmement complexes qu'aborde DocBook. Il est autant respecté dans la communauté DocBook que l'est Linus Torvalds dans le monde Linux.

libxslt est une bibliothèque C qui interprète XSLT et applique les feuilles de style aux documents XML. Elle comprend un utilitaire, **xsltproc**, qui peut être utilisé comme outil de mise en forme XML. Son code a été écrit par Daniel Veillard sous les auspices du projet GNOME, mais ne requiert aucun code GNOME pour fonctionner. J'ai entendu dire qu'elle était extraordinairement plus rapide que ses alternatives Java, ce qui n'est pas une affirmation très surprenante.

xmlto est l'interface utilisateur de la chaîne logicielle intégrée à Red Hat. Elle est écrite et maintenue par Tim Waugh.

Saxon et Xalan sont des programmes Java qui interprètent XSLT. Saxon semble être conçu pour fonctionner sous Windows. Xalan fait partie du projet XML Apache et fonctionne directement sous Linux et BSD. Il est conçu pour fonctionner avec FOP.

FOP traduit des objets de mise en forme XML (FO) en PDF. Il fait partie du projet XML Apache et est conçu pour fonctionner avec Xalan.

8. Outils de migration

Le deuxième plus gros problème de DocBook est l'effort nécessaire pour convertir les anciens balisages de présentation en balisage DocBook. Les êtres humains sont généralement capables de convertir automatiquement la présentation d'un document en une structure logique, parce qu'ils sont, par exemple, capables de distinguer selon le contexte les cas où l'italique est utilisé comme moyen d'accentuation de ceux où il signifie autre chose, comme le fait que la phrase soit en langue étrangère.

D'une façon ou d'une autre, il est nécessaire rendre explicite ce type de distinctions lors de la conversion de documents vers le format DocBook. Parfois elles sont présentes dans l'ancien balisage mais ce n'est en général pas le cas et l'information de structure manquante doit, soit être déduite par une heuristique intelligente, soit être ajoutée par un être humain.

Voici un résumé de l'état des outils de conversion à partir de divers autres formats :

GNU Texinfo

> La *Free Software Foundation* s'est fixé comme politique de permettre d'utiliser DocBook comme format d'échange. Texinfo est suffisamment structuré pour permettre une conversion automatique satisfaisante, et les versions 4.x de **makeinfo** proposent une option `--docbook` qui produit directement du format DocBook. Vous trouverez plus d'information sur la page du projet makeinfo.

POD

> Il existe un module POD::DocBook qui traduit le balisage POD (*Plain Old Documentation*) en DocBook. Il prétend traduire l'intégralité des balises POD sauf la balise italique L<>. La page de manuel précise également « Il n'est pas possible d'utiliser de listes imbriquées pour une sortie en DocBook » et fait remarquer que le module a été intensivement testé.

LaTeX

> LaTeX est (essentiellement) un langage de macros de balisage structurel construit au-dessus de l'outil de mise en forme TeX. Il existe un projet appelé TeX4ht qui (selon l'auteur de PassiveTeX) est capable de produire du code DocBook à partir de LaTeX.

Pages de manuels et balisages basées sur troff

> On considère généralement qu'il s'agit du problème de conversion le plus important et le plus désagréable. Et en effet le balisage de base de troff(1) possède un niveau de présentation trop bas pour que des outils de conversions puissent apporter quelque aide que ce soit. Cependant la situation s'éclaircit significativement si l'on considère les traductions de documents écrits avec des paquets de macros comme man(7). Ceux-ci sont suffisamment structurés pour permettre une traduction automatique.
>
> J'ai moi-même écrit un outil pour ce faire car je n'en trouvé aucun capable de faire proprement le travail (et également parce que le problème est intéressant). Il s'appelle doclifter. Il convertit au format SGML ou XML DocBook les macros man(7), mdoc(7), ms(7) ou me(7). Consultez la documentation pour plus d'informations.

9. Outils d'édition

Il nous manque actuellement est un bon éditeur structurel libre pour les documents SGML et XML.

Le projet Conglomerate vise spécifiquement à fournir un bon éditeur DocBook. Début 2004, ce programme était toujours en version alpha.

Le projet MlView est un éditeur XML généraliste. Début 2004, il n'était pas assez documenté et semblait être en version alpha.

LyX est un éditeur de texte graphique utilisant LaTeX pour l'impression et permettant l'édition structurelle du balisage LaTeX. Il existe un paquet LaTeX qui produit du DocBook et un guide qui décrit comment écrire du SGML et du XML en utilisant LyX.

GeTox, l'éditeur XML de GNOME, vise les utilisateurs ayant un profil non technique. Malheureusement, ce logiciel est toujours en version alpha (depuis août 2001) et est plus un prototype qu'un outil utilisable. En outre, l'équipe du projet ne semble pas très active. Le site du projet n'a pas été mis à jour entre mai 2001 et janvier 2006 (date de publication de la version française de ce guide).

GNU TeXMacs est un projet d'éditeur adapté à la rédaction de documents techniques et mathématiques, et permettant l'affichage de formules. La version 1.0 a été publiée en avril 2002. Les développeurs prévoient une compatibilité XML à l'avenir, mais celle-ci n'est pas encore réalisée.

ThotBook est un projet d'éditeur graphique pour DocBook basé sur la boîte à outils Thot. Il est vraisemblablement moribond car la page web n'a pas été mise à jour entre novembre 2001 et février 2006 (date de publication de la version française de ce guide).

La plupart des gens continuent à écrire directement les balises à la main dans vi ou emacs.

10. Trucs et astuces

Il est possible de produire un index en incluant une balise `<index/>` vide à l'endroit du document où vous voulez qu'il apparaisse. Gardez en mémoire que, début 2004, cette fonctionnalité était toujours assez fruste. Elle ne fusionnait pas les intervalles et le rendu au format PostScript n'était pas encore d'une qualité suffisante pour une utilisation sérieuse.

Cet espace est réservé à d'autres trucs et astuces.

11. Pratiques et standards apparentés

Les outils d'édition et de mise en forme du balisage DocBook sont en train d'émerger, bien que lentement. Mais DocBook en soi est un moyen et non une fin. Nous aurons besoin d'autres standards en plus de DocBook pour atteindre l'objectif de bases de documents interrogeables que j'ai évoqué au début de ce document. Les deux enjeux principaux sont le catalogage de document et les méta-données.

Le projet Scrollkeeper a pour objectif de satisfaire ce besoin. Il fournit des points d'entrées utilisables par les scripts d'installation et de désinstallation de paquets pour enregistrer et désenregistrer leurs documentations dans une base de données commune partagée et interrogeable.

Scrollkeeper utilise le format Open Metadata Format. Il s'agit d'un standard d'indexation des documentations libres similaire à un système de catalogue sur fiche de bibliothèque. L'idée est d'offrir des fonctionnalités de recherches avancées utilisant aussi bien les méta-données du catalogue que le texte de la documentation elle-même.

12. SGML et les outils SGML

Dans les sections précédentes j'ai laissé de coté une bonne partie de l'historique de DocBook. XML a un grand frère, SGML ou *Standard Generalized Markup Language*.

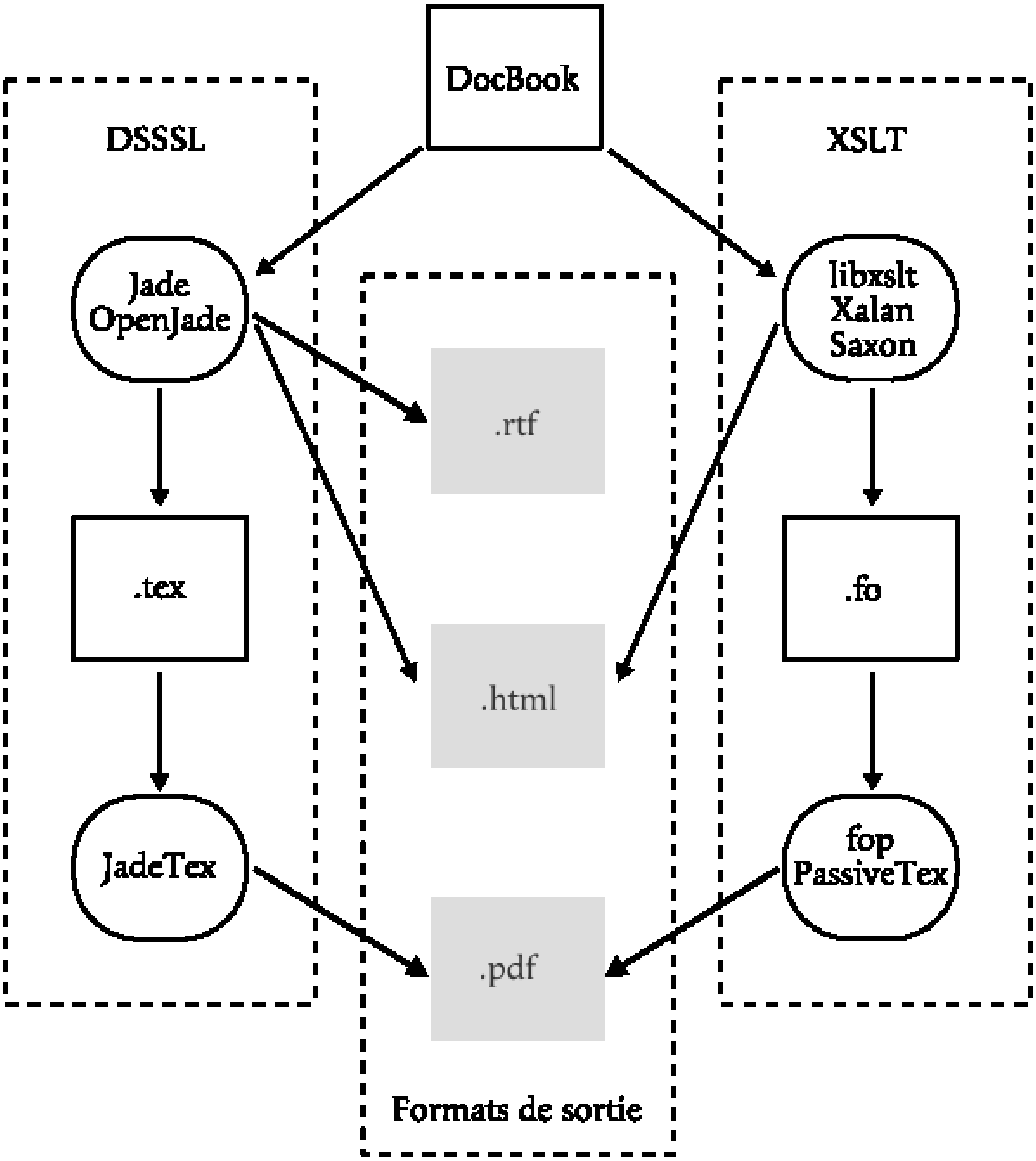
Jusqu'en mi-2002, aucune discussion au sujet de DocBook n'aurait été complète sans une longue excursion dans SGML, les différences entre SGML et XML et des descriptions détaillées de la chaîne logicielle SGML DocBook. La vie est désormais plus simple. Une chaîne logicielle XML DocBook libre est disponible, elle fonctionne aussi bien que ne l'a jamais fait la chaîne SGML et est beaucoup plus facile à utiliser. Si vous ne pensez jamais avoir à traiter d'anciens documents SGML DocBook, vous pouvez sauter la suite de cette section.

12.1. DocBook SGML

DocBook était à l'origine une application de SGML et il existait une chaîne logicielle DocBook basée sur SGML, qui est désormais moribonde. Il existe des différences mineures entre la DTD DocBook SGML et la DTD DocBook XML que nous pouvons ignorer ici. La seule qui soit visible par les utilisateurs est le fait que les balises SGML vides ne nécessitent pas d'avoir une barre oblique ajoutée avant le > de fermeture (la barre oblique obligatoire signifie que les analyseurs XML peuvent être beaucoup plus simples car ils n'ont pas à connaître la DTD pour savoir quelles balises d'ouverture nécessitent des balises de fermeture).

Jusqu'à la version 4.01 (avant XHTML), HTML était une application de SGML. TEI était à l'origine également une application de SGML. Les équipes en charge de ces deux DTD sont passées à XML pour la même raison que les développeurs de DocBook — c'est radicalement plus simple. SGML était extrêmement complexe et s'est donc avéré impossible à gérer. La spécification comportait 150 pages très denses et il n'a jamais été vérifié qu'un logiciel l'ait entièrement mis en œuvre.

Le schéma de chaîne logicielle que j'ai donné plus tôt était simplifié car il montrait uniquement la chaîne XML. Voici la version historiquement correcte :



La chaîne logicielle DSSSL servait à traiter le SGML DocBook. Dans cette chaîne, un document au format DocBook est traité par un des deux moteurs de feuilles de style (Jade ou OpenJade). Il est ainsi transformé en balisage de macros TeX qui était convertit en DVI par un paquet appelé JadeTex pour être finalement transformé en PostScript.

12.2. Outils SGML

Le projet docbook-tools fournit des outils libres pour convertir du SGML DocBook en HTML, PostScript et dans d'autres formats. Ce paquet est intégré à la distribution Red Hat et à d'autres distributions Linux. Il est maintenu par Mark Galassi.

Jade est un moteur utilisé pour appliquer des feuilles de style DSSSL à des documents SGML. Il est maintenu par James Clark.

OpenJade est un projet communautaire entrepris parce que ses créateurs pensaient que la maintenance de Jade réalisée par James Clark n'était pas satisfaisante. Les programmes de docbook-tools utilisent OpenJade.

PassiveTeX est le paquet de macros LaTeX utilisé par xmlto pour produire du DVI à partir du XML DocBook. JadeTex est le paquet de macros LaTeX utilisé par OpenJade pour produire du DVI à partir du SGML DocBook.

12.3. Pourquoi le SGML DocBook est mort

La chaîne logicielle DSSSL, si l'on considère les nouveaux développements, est, en pratique, morte. La chaîne XSLT a atteint mi-2002 une qualité suffisante pour les environnements de production. Une version fonctionnelle de cette chaîne est intégrée à la distribution Red Hat 7.3. C'est sur cette chaîne que les développeurs DocBook concentrent la plupart de leurs efforts.

La raison du passage à XML est triple. Premièrement, SGML s'est avéré être trop compliqué à utiliser ; deuxièmement, DSSSL s'est avéré trop compliqué à appliquer et enfin des parties significatives de la chaîne DSSSL se sont avérées être fragiles et trop mal conçues.

Apparenté à SGML, XML possède un nombre de fonctionnalités réduit, ce qui est suffisant dans la plupart des cas tout en simplifiant grandement sa compréhension et la mise en œuvre d'analyseurs. Les outils de traitement SGML (comme les analyseurs de validation) doivent mettre en œuvre bon nombre de fonctionnalités non utilisées par DocBook et les autres systèmes de balisage de textes. Le fait d'enlever ces fonctionnalités a rendu XML plus simple et les outils de traitement XML plus rapides.

Le langage utilisé pour décrire les DTD SGML est suffisamment épineux et rébarbatif pour que l'écriture de DTD SGML relève du domaine de la science occulte. Les DTD XML, d'un autre coté, peuvent être décrites comme un dialecte d'XML lui-même. Un langage de DTD séparé n'a donc pas lieu d'être. Une description XML d'une DTD XML est appelée un *schéma*. L'usage du terme DTD lui-même va probablement se perdre au fur et à mesure de la standardisation des schémas.

Cependant la chaîne logicielle DSSSL est surtout morte à cause de DSSSL lui-même : le langage de description des feuilles de style de SGML était trop obscur pour la plupart des êtres humains ce qui a rendu les feuilles de style trop difficiles à écrire et à modifier[3 [p 14]].

Les défenseurs d'XML aiment résumer tous ces changements ainsi : « XML : ça a bon goût et c'est facile à digérer. »

12.4. SGML-Tools

SGML-Tools était le nom d'une DTD utilisée par le Projet de documentation Linux (LDP), développée il y a quelques années alors que la chaîne logicielle DocBook n'existait pas. Le balisage de SGML-Tools était plus simple mais aussi beaucoup moins souple que DocBook. La chaîne logicielle SGML-Tool d'origine (outil de mise en forme, DTD et feuilles de style) est morte depuis déjà un certain temps mais son successeur, appelé SGML-tools Lite est toujours mis à jour.

Le Projet de documentation Linux a progressivement abandonné SGML-Tools au profit de DocBook, mais il est toujours possible que vous ayez à reprendre de vieux guides pratiques. Ils sont reconnaissables par leur entête d’identification « <!doctype linuxdoc system> ». Si jamais cela vous arrive, convertissez cette chose au format XML DocBook et enterrez rapidement la vieille version.

13. Références

Une des choses qui compliquent l’apprentissage de DocBook est le fait que les sites sur le sujet ont tendance à submerger les débutants de listes de standards du W3C, d’exercices volumineux sur la théologie du balisage et de terminologie abstraite.

« *Take My Advice: Don’t Learn XML* » (« Suivez mon conseil, n’apprenez pas XML ») de Michael Smith sonde le monde d’XML depuis un angle similaire à celui de ce document.

« DocBook: The Definitive Guide » Norman Walsh est disponible en version papier et sur le web. C’est en fait la référence qui fait autorité mais c’est un désastre en tant que document d’introduction ou d’apprentissage. Lisez plutôt ceci :

« *Writing Documentation Using DocBook: A Crash Course* ». C’est un excellent guide d’apprentissage.

Il existe une excellente « FAQ DocBook (en anglais) » contenant de nombreuses informations sur la personnalisation du style du HTML produit. Il existe également un wiki DocBook (en anglais).

Si vous écrivez pour le Projet de documentation Linux, lisez le « *LDP Author Guide* ».

La meilleure introduction à SGML et XML, que j’ai personnellement lue d’une traite, est celle de David Megginson : « *Structuring XML Documents* » (Prentice-Hall, ISBN : 0-13-642299-3).

Au sujet d’XML, « XML en concentré » de W. Scott Means et Elliotte « Rusty » Harold, est très bien.

« XML guide de l’utilisateur » semble être un ouvrage de référence assez approfondi sur XML et les standards associés (dont notamment Formatting Objects).

Enfin, le site « *The XML Cover Pages* » vous entraînera dans la jungle des standards XML, si c’est vraiment ce que vous voulez.

[1 [p 3]] N.D.T. : les fautes sont volontaires. Pour comprendre cette référence, jetez un œil à l’article Wikipédia correspondant.

[2 [p 7]] PDF signifie Format Portable de Document — *Portable Document Format* dans la langue de Shakespeare.

[3 [p 13]] Il s’agissait d’un dialecte de Scheme. Votre humble auteur, adepte de LISP depuis des lustres, est pris d’effarement à l’idée que cela puisse en faire fuir certains