Formation en Calcul Scientifique - LEM2I

–

Doxygen, génération automatique de documentation

Violaine Louvet [1]

[1] Institut Camille jordan - CNRS

12-14/12/2011

Introduction

Pourquoi générer automatiquement la documentation ?

- La seule source d'information absolument juste est le code.
- La rédaction de la documentation technique est laborieuse et complexe.
- Faciliter la maintenance, le développement d'un code écrit seul et surtout à plusieurs .
- La documentation est écrite dans le code, et il est donc relativement facile de la tenir à jour.

Comment ça marche ?

- Extraction de l'information à partir du code source et d'autres données laissées à la responsabilité du développeur
- Tient compte de la syntaxe et de la structure du langage du programme ainsi que des commentaires associés

Différents outils

http://en.wikipedia.org/wiki/Comparison_of_documentation_generators

Doxygen

- Langages supportés : C++, C, Java, Objective-C, Python, IDL, Fortran, VHDL, PHP, C# ...
- Basé sur un ensemble de balises à ajouter dans les sources
- Formats de sorties : Différents formats de sorties : RTF (MS-Word), PostScript, PDF avec liens hypertexte, HTML (compressé ou pas), Unix, Man pages, LaTeX
- Systèmes d'exploitation : la plupart des systèmes Unix, y compris Mac OS X, ainsi que Windows
- Licence GPL

Sommaire

Sommaire

Règles de base

- Tout la documentation est générée à partir des commentaires du code selon quelques règles simples.
- Afin de différencier dans un fichier les commentaires à l'intention de Doxygen de ceux qui ne sont pas à prendre en compte, Doxygen possède sa propre norme de documentation :
  - « *!>* » ou « *!<* » débutent un commentaire doxygen
  - « *!>* » ou « *!!* » permettent de poursuivre le commentaire doxygen sur plusieurs lignes.
- Les formes de commentaires conventionnelles ne sont pas pris en compte par Doxygen.
- L'emplacement par défaut pour les commentaires est avant la structure à documenter.
  - Le commentaire peut être placé à un autre endroit, dans ce cas, il faut préciser par un mot-clé le type de structure documentée

Exemple

!> Build the matrix
!! for LU decomposition

Sommaire

Balises

Principe

- Afin de préciser et de donner du sens aux commentaires, chaque information essentielle dispose d'une balise pour en préciser la nature.
- Une balise peut s'écrire \balise ou @balise.

Mise en forme

- \a : mise en italique du mot suivant. Pour plusieurs mot on peut utiliser * ... *
- \b : mise en gras du mot suivant. Pour plusieurs mot on peut utiliser *<tt> ... </tt>*
- Listes :

```
!> Liste de fonctionnalités :
!!  - solveur linéaire
!!       -# gradient conjugué
!!       -# décomposition LU
!!  - sorties graphiques
!!       -# VTK
!!       -# gnuplot
```

Documenter un fichier

```
!> \file      Nom du fichier
!! \author    Nom de l'Auteur
!! \version   1.0
!! \date      14 decembre 2011
!! \brief     Blabla resume
!!
!! \details   Blabla beaucoup plus
!!            detaille
```

A vous

- Reprendre le programme du cours d'optimisation et documenter le fichier.

Documenter un type

```
!> Nom de la structure
!! @param param1 description du param1
!! @param param2 description du param2
!! @param param3 description du param3
```

A vous

- Ajouter un type *Matrix* au programme
- Le documenter

```
type Matrix
   integer    :: n
   real(kind=8), pointer, dimension(:,:) :: val
end type Matrix
```

Documenter une routine

```
!> Petite description de ce que fait la routine
!! @param [in] arg1 description de l'argument 1
!! @param [in] arg2 description de l'argument 2
!! @param [in,out] arg3 description de l'argument 3
!! @return ret description de la variable de retour
```

A vous

- Documenter les routines du code

Autres possibilités

- Utiliser des tags HTML dans la documentation :

```
!! <a href=''../../README''>Readme file </a>
```

- Intégrer des liens :

```
!! \see #routine_interne
!! \see fichier#routine
```

- Ajouter des directives *TODO* ou *Bug* :

```
!! \bug
!! Corriger les indices de boucles
!! \todo
!! Ajouter un type vector
```

A vous

- Ajouter une tâche *TODO*

Page principale

Doxygen fournit la possibilité de générer la page principale de la documentation.

```
!! \mainpage The optim program
!!
!! \section intro Introduction
!!     blablabla
!! \section optims Optimisations
!!    \subsection loops Les boucles
!!       blablabla
!!    \subsection cstes Les constantes
!!       blablabla
!!       lien avec les boucles : voir \ref loops
!! \section install Installation
!!    Voir le fichier <a href=''../../INSTALL''>INSTALL</a>
```

A vous

- Intégrer la documentation de la page principale

Sommaire

Sommaire

Fichier de configuration

- Comprend les diverses options pour la génération de documentation
- Peut être généré comme un patron (toutes les options prennent leurs valeurs par défaut et ils sont précédés d'un commentaire explicatif) :

```
\$> Doxygen -g configFile
```

Sections du fichier de configuration

- options du projet
- options de compilation
- options de traitement des fichiers source
- options de sortie
- options pour les graphes (dot)

Description du fichier de configuration

Options du projet

- *PROJECT_NAME* : nom du projet
- *PROJECT_NUMBER* : version
- *OUTPUT_DIRECTORY* : répertoire dans lequel sera générée la documentation
- *OUTPUT_LANGUAGE* : langage dans lequel sera généré la documentation
- *OPTIMIZE_FOR_FORTRAN* : génération de documentation en prenant en compte davantage les spécificités Fortran

Options de compilation

- *EXTRACT_ALL* : extrait toute la documentation. Doxygen considère que tous les fichiers sont documentés
- *SOURCE_BROWSER* : intègre des liens directs vers les sources dans la documentation

Description du fichier de configuration

Options de traitement des fichiers source

- *INPUT* : répertoires ou fichiers contenant les codes source à analyser
- *RECURSIVE* : l'analyse du répertoire source se fera de façon récursive. Les sources contenus dans les sous-répertoires seront analysée

Options de sortie

- *GENERATE_HTML* : génère une documentation html
- *GENERATE_LATEX* : génère une documentation LaTeX
- *GENERATE_RTF* : génère une documentation rtf
- *GENERATE_MAN* : génère une documentation man pages

Description du fichier de configuration

Options pour les graphes

- *CLASS_DIAGRAMS* : génération de diagrammes dans la documentation. Pour cela, Doxygen peut s'appuyer sur l'outil *dot* du logiciel *Graphviz*
- *HAVE_DOT* : indique si l'outil *dot* est installé
- *CALL_GRAPH* : génération de graphes de dépendances des appels de fonctions

A vous

- Générer le fichier de configuration pour *Doxygen*
- Le modifier
- Construire la documentation : *doxygen*

Sommaire

Assistant Wizard

Démo de l'outil *doxywizard*

Sommaire

Structure du répertoire de documentation