Table des matières

L'auto-documentation

Besoins

[La morale dit qu'il faut documenter ses programmes :-]

Première question: pourquoi documenter ?

- ➢ mon employeur ou le client la demande
- ➢ je travaille dans une équipe de développement
- ➢ pour mieux comprendre mon code lorsque je retravaille dessus
- ✗ on se sent mieux avec !

Deuxième question: pourquoi utiliser un système de documentation automatique ?

- ➢ elle sera toujours à jour
- ➢ réutiliser les commentaires déjà insérés dans les sources
- ➢ choisir le format de sortie (HTML, PDF, RTF, ...)
- ➢ obtenir des références croisées permettant une meilleure navigabilité
- ➢ récupérer les parties importantes en signification des sources
- ✗ on travaille moins !

Troisième question: pourquoi doxygen ?

- ➢ il correspond aux besoins exprimés ci-dessus
- ➢ il est *free* (gratuit et libre)
- ➢ il est simple, rapide et configurable
- ✗ on m'oblige à l'utiliser !

Outils non commerciaux

AutoDOC, Autoduck, Cocoon, CcDoc, CppDoc, Cxref, cxxwrap, Cxx2HTML, C2HTML, Doc++, DocClass, Epydoc, gtk-doc, HappyDoc, **HeaderDoc**, HTMLgen, HyperSQL, **Javadoc**, **KDoc**, Natural Docs, Perceps, phpDocumentor, PHPDoc, ReThree-C++, RoboDoc, ScanDoc, Synopsis, Tydoc, VBDOX, ...

Outils commerciaux

DocBuilder, DocJet, Doc-o-matic, ObjectManual, Together, CC-Rider, VBXC, ...

Doxygen

Introduction

Doxygen est un outil qui permet de générer automatiquement une documentation formatée, navigable et imprimable à partir de fichiers sources spécialement préparés à son effet.

Doxygen supporte les langages C, C++, Java, et IDL en utilisant deux styles différents de documentation.

Doxygen est capable de produire des documentations en HTML, LATEX, XML et RTF (*Rich Text Format*). En utilisant des outils additionnels, on pourra aussi fournir une documentation dans beaucoup d'autres formats tel que : PDF, pages *man*, PostScript, *compressed* HTML (HTML *Help* .chm) et Micro$oft *Word*.

Caractéristiques

- portable (disponible pour les plate-formes Unix, Window$ et MacOS X).
- compatible avec Javadoc, Qt-Doc et KDOC.
- reconnaissance automatique et génération de références croisées
- utilisation de la coloration syntaxique
- génération automatique de diagrammes de classe
- organisation par groupe pour des documentations spécialisées
- etc ...

Utilisation

En ligne de commande

1 . Générer le fichier de configuration : **`doxygen -g filename`** (ou **`filename.doxy`**)

2 . Editer le fichier de configuration : **`vim filename`** et adapter les options en fonction du résultat désiré, par exemple:

```
PROJECT_NAME = Mon Projet
PROJECT_NUMBER = 1.0
OUTPUT_DIRECTORY = /home/tv/tv/tmp/monProjet/doc/
OUTPUT_LANGUAGE = French
EXTRACT_ALL = YES
SOURCE_BROWSER = YES
GENERATE_HTML = YES
INPUT = /home/tv/tv/tmp/monProjet/src/
FILE_PATTERNS = *.c
...
```

Remarque: l'ensemble des options de configuration sont accessibles à l'adresse suivante http://www.stack.nl/~dimitri/doxygen/config.html ou en local dans /usr/share/doxygen-1.2.18/

3 . Générer la documentation: **`doxyden filename`**

En version GUI (*Graphic User Interface*)

Utiliser **`doxywizard`**

Les différents styles

Doxygen reconnaît deux styles de commentaires : *Javadoc-style* et *Qt-style*.

Javadoc-style

```
/**
 * Fonction documentation.
 * @param x  Le paramètre.
 * @return  La valeur retournée .
 * @see autreFunction()
 */

/** simple ligne de documentation. */
///  simple ligne de documentation.

int a; ///< la documentation est après
```

Qt-style

```
/*!
 Fonction documentation.
 \fn laFoction(int x)
 \brief une brève description
 \param x Le paramètre.
 \return La valeur retournée.
 \sa autreFunction()
*/

/*! simple ligne de documentation. */
//! simple ligne de documentation.

char b; //!<  la documentation est après (Qt-style)
```

Les commandes spéciales

Doxygen interprète les commandes spéciales placées dans le code source afin de documenter celui-ci. Il existe de nombreuses commandes spéciales, les plus utilisées sont présentées dans les exemples ci-dessous.

L'ensemble des commandes spéciales: http://www.stack.nl/~dimitri/doxygen/commands.html ou en local dans /usr/share/doc/doxygen-1.2.18/

Les listes

Doxygen permet de créer des listes (simples, numérotées ou directement en HTML), par exemple :

```
/*! Une liste
 * - item1
 * - item2
 *   -# sous item21
 *   -# sous item22
 * - item3
 */
```

Exemples

Les sources

```
/*!
 * \file jeu.c
 * \brief Gestion du jeu
 * \author \a tv
 * \version 1.0
 * \date \b 2005
 *
 * \bug fin de partie non gérée
 * \warning certaines fonctions ne sont pas implémentés
 * \todo ::verifierPlateau(int *plateau)
 * \test ...
 *
 * Une description plus détaillée ...
 *
 */
```

Remarques
Les commandes spéciales \bug, \warning, \todo et \test généreront des listes séparées.
Certaines commandes spéciales (\author, \date, \file ...) peuvent être combinées avec un gestionnaire de versions comme **RCS/CVS** ($Author$, $Date$, ...).

Les fonctions

```
/*!
 * \fn verifierPlateau(int *plateau)
 * \brief vérifie le plateau de jeu à la recherche d'un gagnant ou d'un
match nul
 * \param plateau (int *) pointeur sur le tableau de jeu
 * \return gagne (int) état de la partie en cours
 * \retval gagne [::JOUEUR | ::ORDI | ::FINI]
 *
 * \todo la recherche de l'état de la partie en cours [::JOUEUR | ::ORDI
| ::FINI]
 * \warning non implémentée (renvoie toujours 0)
 * \test
 * - le joueur ou l'ordi a aligné 3 pions verticalement, horizontalement ou
diagonalement \n
 * - partie finie sans gagnant -> match nul
 *
 */
int verifierPlateau(int *plateau)
{ ... }
/*!
 * \fn jouerOrdi(int *plateau)
 * \brief permet à l'ordinateur de jouer un pion
 * \param plateau (int *) pointeur sur le tableau de jeu
 * \return joue (int) indique si l'ordinateur a pu jouer
 * \retval joue [0 | 1]
 * \todo intégrer l'IA
 * \sa jouerPosition(int *plateau, int position)
 */
int jouerOrdi(int *plateau)
{ ... }
```

Remarques: voir aussi \var et en **C++/Java** \class, \exception, \throw, \package, \interface ...

Les définitions

```
/*! \def LIBRE identifie une case de libre */
#define LIBRE     0
/*! \def JOUEUR identifie le joueur */
#define JOUEUR    1
/*! \def ORDI identifie l'ordi */
#define ORDI      2
/*! \def FINI état de la partie */
#define FINI      3
```

Remarques: voir aussi \typedef, \enum, \union, \struct ...

Les autres pages de documentation

Les commandes spéciales permettent de créer des pages (\mainpage et \page) de documentation structurées en sections (\section) et sous-sections (\subsection) contenant des liens de référence croisée (\ref) :

```
/*! \mainpage Page principale du projet morpion
 *
 * \section intro Introduction
 *
 * Jeu de morpion en langage C \n
 * L'objectif est de jouer une partie contre l'ordinateur et de le battre !
 *
 *
 * \section tdm Table des matières
 * - \ref page1
 * - \ref licence
 * - \ref about1
 */

/*! \page page1 Documentation
 * -# \ref Installation
 * -# \ref Usage
 * -# \ref Configuration
 *
 * Voir aussi la page \ref about1.
 * \subsection Installation Manuel d'installation
 * - \c make \c dep
 * - \c make \c all
 * - \c make \c install
 *
 * \subsection Usage Manuel d'utilisation
 * - Taper: \c morpion et le jeu démarre
 * - Utiliser le pavé numérique pour jouer un pion sur le plateau de jeu
 *
 * Le programme morpion est sous \ref licence.
 *
 * \subsection Configuration Configuration requise
 * - Linux et la libc.so.6
*/

/*! \page about1 A propos
 * \author \a tv
 * \version 1.0
 * \date \b 2005
 */

/*! \page licence Licence GPL
```

```
*
*   This program is free software; you can redistribute it and/or modify
*   it under the terms of the GNU General Public License as published by
*   the Free Software Foundation; either version 2 of the License, or
*   (at your option) any later version.
*
*   This program is distributed in the hope that it will be useful,
*   but WITHOUT ANY WARRANTY; without even the implied warranty of
*   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*   GNU General Public License for more details.
*
*   You should have received a copy of the GNU General Public License
*   along with this program; if not, write to the Free Software
*   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
 */
```

Travaux pratiques

Séquence 1

1 . Récupérer l'archive du projet **Morpion** et, avec **`doxywizard`**, générer différents types de documentation **HTML** en regardant notamment les options : GENERATE_TREEVIEW, EXTRACT_ALL, GENERATE_xxxLIST (TODO, TEST, ...) ...

2 . Puis générer une documentation au format **RTF** et utiliser *OpenOffice Writer* pour la lire.

Séquence 2

1 . Documenter la partie *ihm* du projet Morpion et générer la documentation associée au format **HTML**.