Doxygen

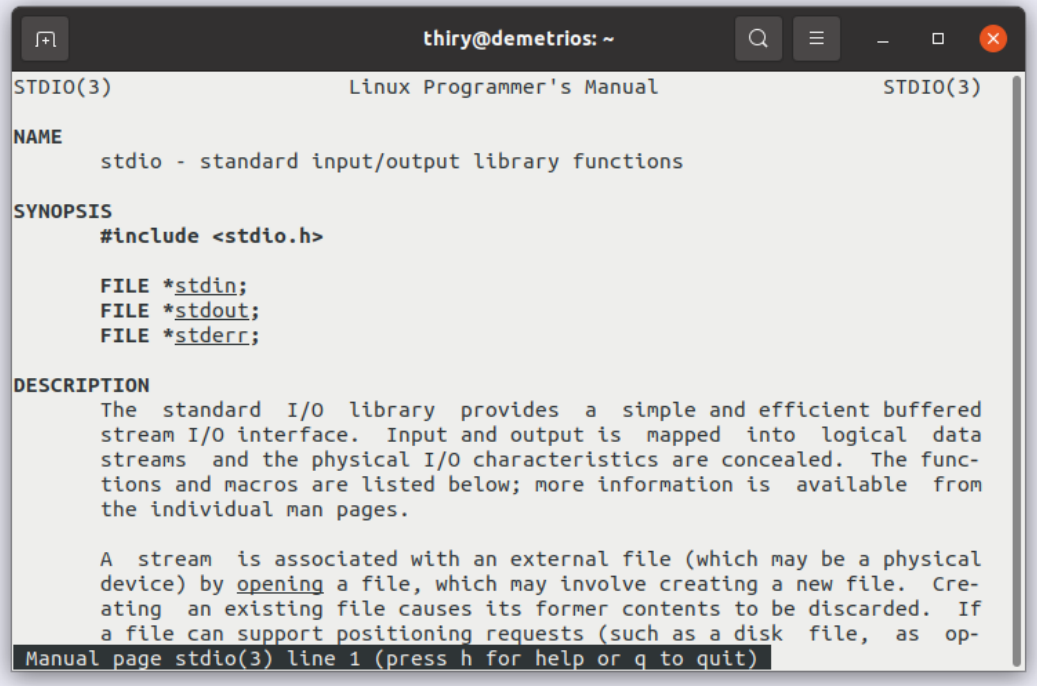
Un générateur de documentation

laurent.thiry@uha.fr

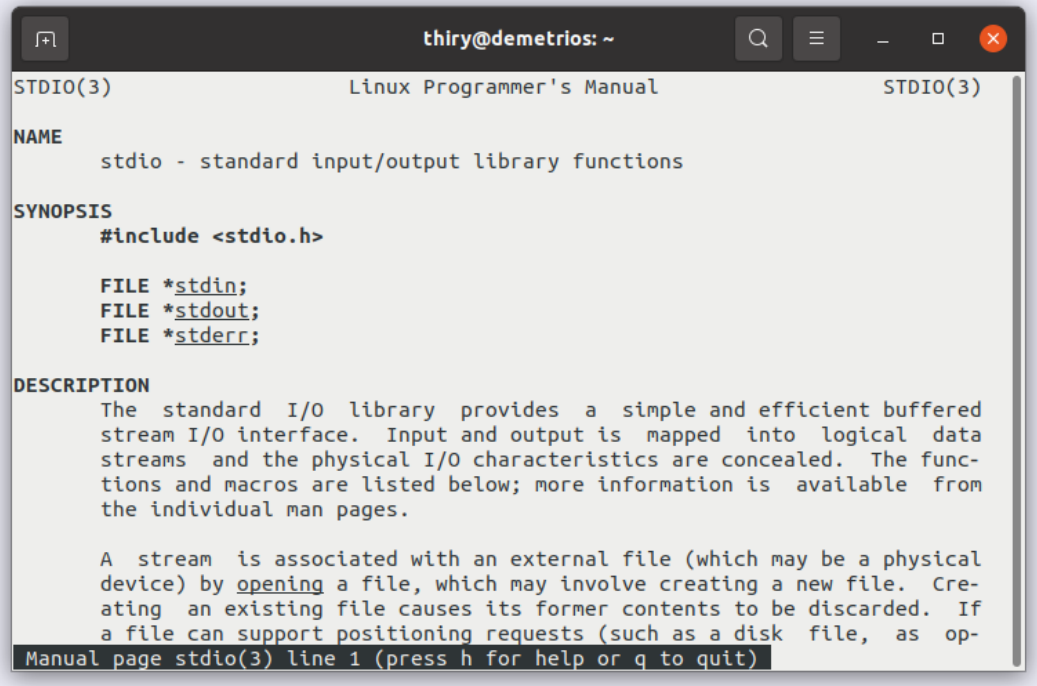
[1] https://www.doxygen.nl/index.html

Motivation: comment profiter de "stdio.h" & co. ?

Motivation: comment profiter de "stdio.h" & co. ?



Motivation: comment profiter de "stdio.h" & co. ?



⇒ Comment faire une "bonne" documentation ?

Principe

1. Utiliser des /*commentaires*/ ... pour chaque élément de code
   - fichiers
   - structures et/ou classes
   - sous-programmes et fonctions
   - ...

Principe

1. Utiliser des /*commentaires*/ ... pour chaque élément de code
   - fichiers
   - structures et/ou classes
   - sous-programmes et fonctions
   - ...
2. Intégrer des \balises spécifiques
   - auteur(s)
   - champ(s)
   - paramètre(s)
   - ...

Par exemple: comment profiter du code suivant ?

```
struct  Vector { int x; int y; };
typedef struct Vector* Vect;

Vect new_Vect(int x, int y);
Vect add(Vect v1, Vect v2);
void print(Vect v);
```

Par exemple: comment profiter du code suivant ?

```
struct  Vector { int x; int y; };
typedef struct Vector* Vect;

Vect new_Vect(int x, int y);
Vect add(Vect v1, Vect v2);
void print(Vect v);
```

- Est-ce que "add" modifie les paramètres ?

Par exemple: comment profiter du code suivant ?

```
struct  Vector { int x; int y; };
typedef struct Vector* Vect;

Vect new_Vect(int x, int y);
Vect add(Vect v1, Vect v2);
void print(Vect v);
```

- Est-ce que "add" modifie les paramètres ?
- Est-ce que "print" fait appel l'imprimante ?
- ...

Avec des commentaires ... c'est plus facilement (ré)utilisable

```
/**
 *  "Vect.h"
 *  v0.1
 */

/**
 *  Vecteur 2D en coordonnées entières.
 */
struct Vector { int x; int y; };
typedef struct Vector* Vect;

/**
 *  Constructeur
 */
Vect new_Vect(int x, int y);
```

En ajoutant des \balises (éléments sémantiques) pour les fichiers:

```
/**
 *  \file Vect.h
 *  \brief Vecteurs 2D avec opérations usuelles (add,print)
 *  \author laurent.thiry@uha.fr
 *  \version 0.1
 *
 *  Exemple de code source documenté avec Doxygen.
 */
```

... pour les structures (ou des classes C++, etc.):

```
/**
 *  \struct Vector Vect.h Vect.c
 *  \brief vecteur 2D avec coordonnées entières (4 octets)
 */
struct Vector {
/*! abscisse */
int x;
/*! ordonnée */
int y;
};
typedef struct Vector* Vect;
```

... pour les programmes:

```
/**
 *  \fn new_Vect(int x, int y)
 *  \brief constructeur
 *  \param x abscisse entière
 *  \return référence sur un nouveau vecteur
 */
Vect new_Vect(int x, int y);
```

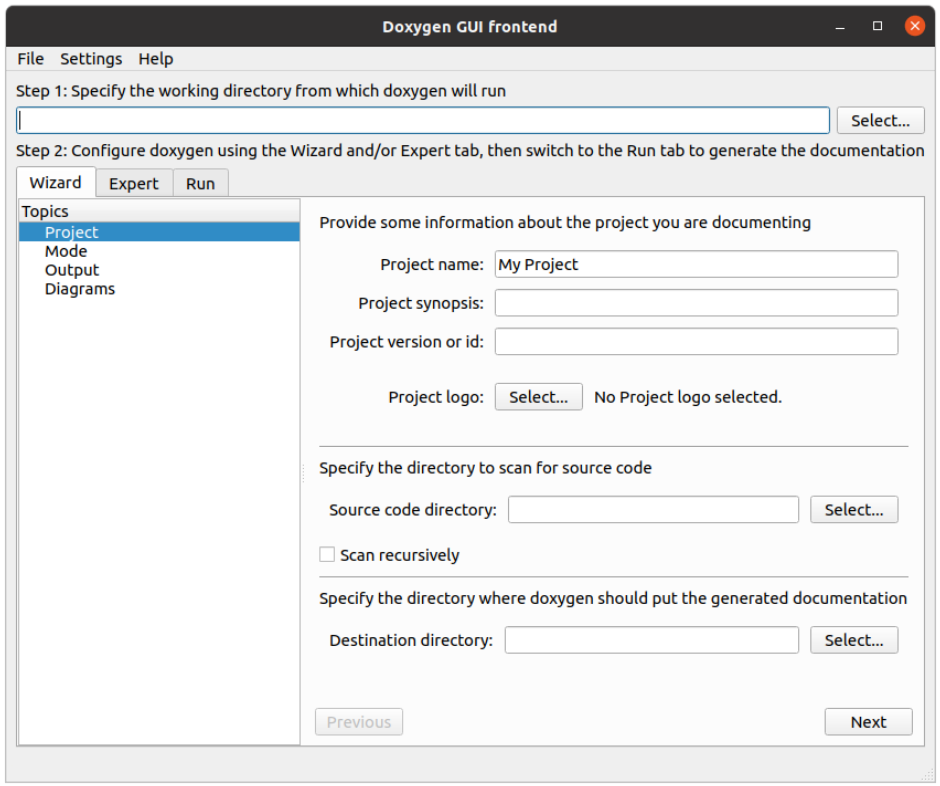
... pour les programmes:

```
/**
 *  \fn new_Vect(int x, int y)
 *  \brief constructeur
 *  \param x abscisse entière
 *  \return référence sur un nouveau vecteur
 */
Vect new_Vect(int x, int y);
```

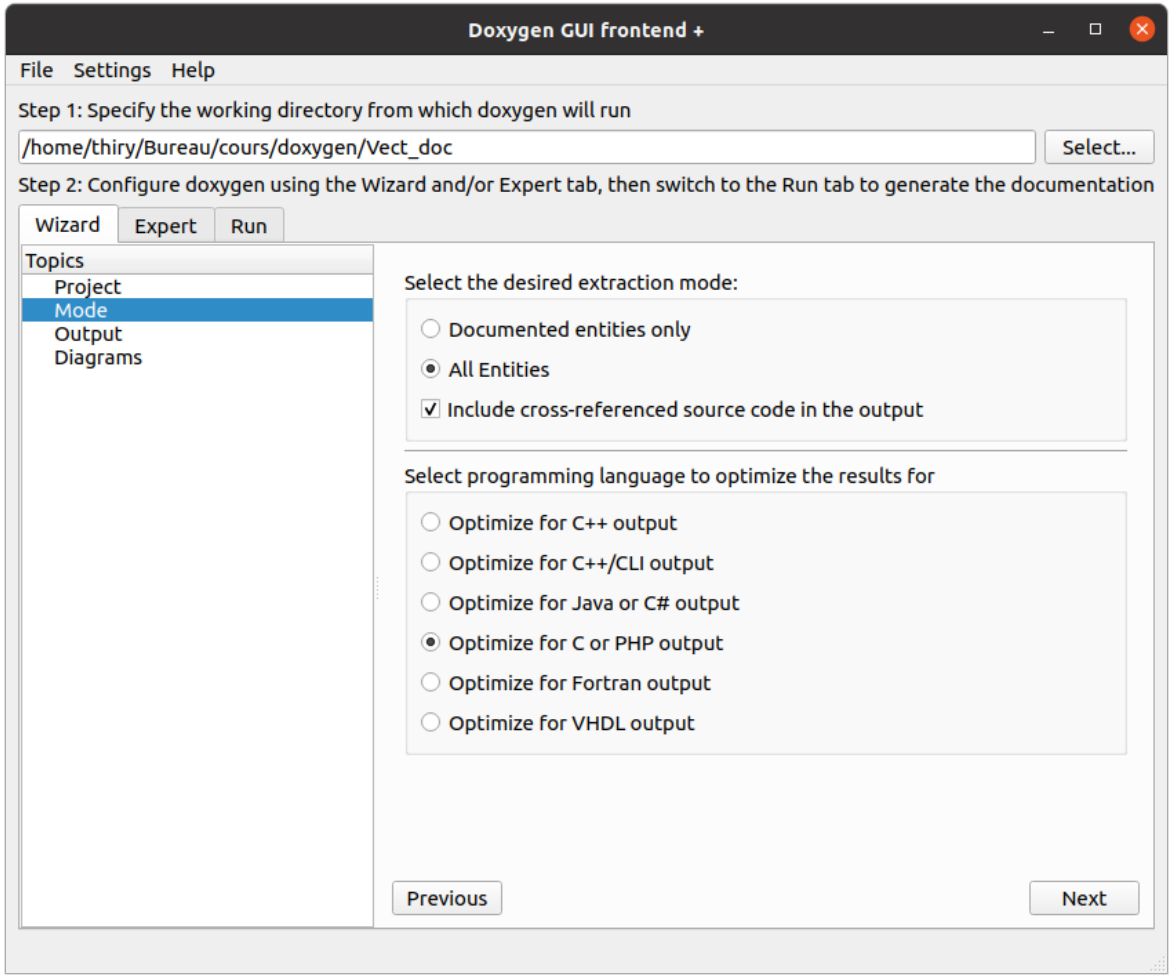
$\Rightarrow$ *Doxygen* permet alors de générer une documentation Html à partir de ces nouvelles informations ! Et plus encore...

```
# Installation
sudo apt install graphviz doxygen doxygen-gui
```

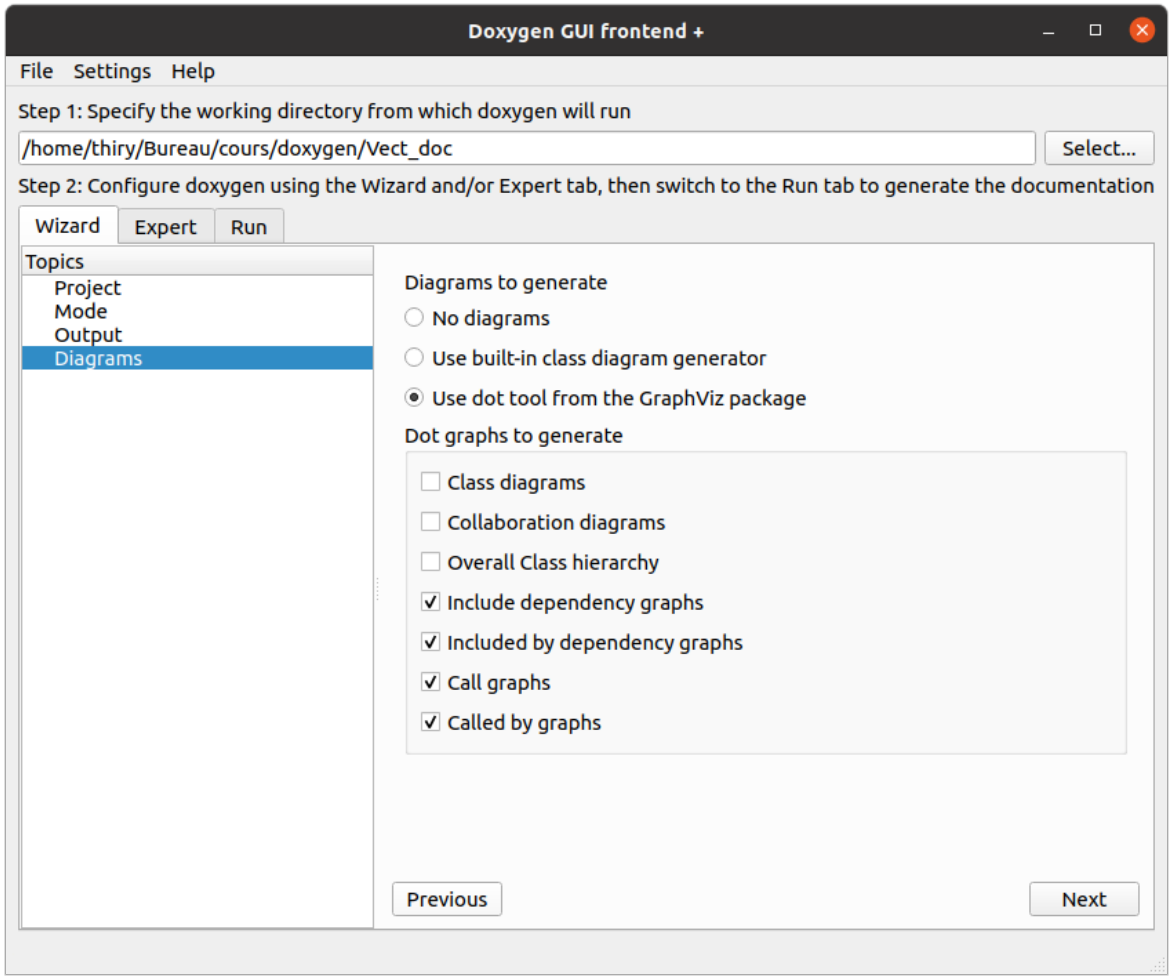
```
# Installation
sudo apt install graphviz doxygen doxygen-gui

# Lancement
doxywizard
```



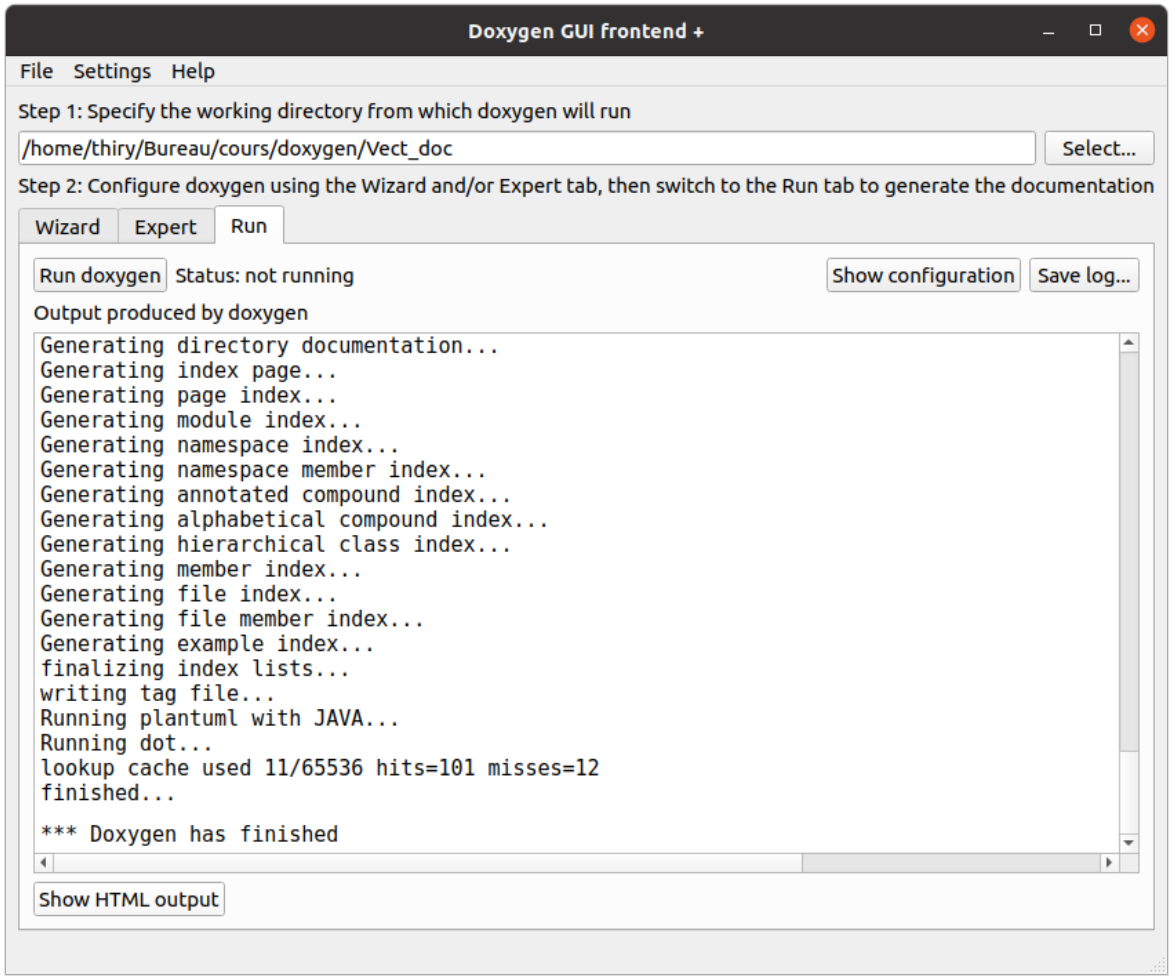
⇒ Prendre du temps pour bien configurer les paramètres pour avoir un meilleur résultat ...



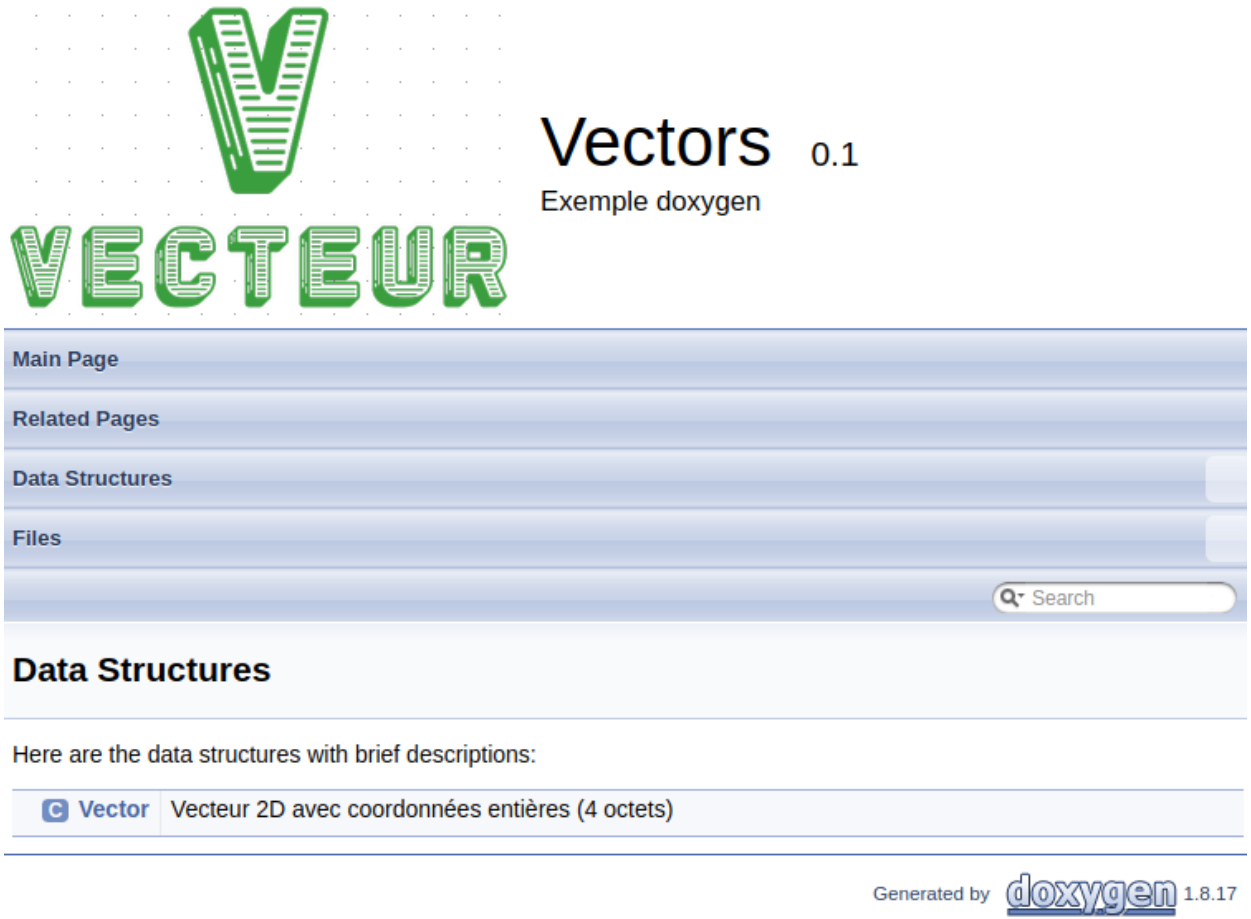
⇒ Doxygen permet de générer le graphe de dépendances !



Enfin, on génère la documentation...



Et voilà !



[1] `file:///home/thiry/Bureau/cours/doxygen/Vect_doc/html/index.html`

Et il y a un peu plus que les commentaires :-)

◆ new_Vect()

```
Vect new_Vect ( int x,
                int y
              )
```

constructeur

Parameters

x abscisse entière

Returns

référence sur un nouveau vecteur

Definition at line 5 of file Vect.c.

Here is the caller graph for this function:

Merci de votre attention

Avez-vous des questions ?