Programmation en C
Cours 8

Licence Maths-Info
Aix-Marseille Université
2011-2012

Valentin Emiya
valentin.emiya@lif.univ-mrs.fr

19 mars 2012

Séances 7 et suivantes

Séances 1 à 6	PARTIEL	Séances 7 et suivantes
Apprentissage du langage Outils de base (compilation, etc.) Pratiques élémentaires		Pratique du C Bonnes pratiques

Objectifs des séances à venir :

- Bonnes pratiques et pratiques courantes pour le développement en C
- Mise en pratique « traitement d'image » (lecture/écriture, transformations/effets)

Séances de TP supplémentaires :

mardis 20/03 et 17/04 à 16h en salle 301

Lundi dernier

Pratique du langage C (1)

* Organisation des fichiers pour créer un logiciel
* Les bibliothèques statiques et dynamiques
  * Principe
  * Syntaxe, sémantique
  * Intérêts et inconvénients
  * Compilation
* Introduction du thème « traitement d'image »

Aujourd'hui

Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image (type, patch)
* Doxygen : outil de documentation du code

Aujourd'hui

Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image (type, patch)
* Doxygen : outil de documentation du code

Pointeurs vs tableaux : *déclaration + allocation mémoire*

= *différences importantes*

Tableaux :

```
#define N 10 ;
double t[10] ;
double u[N] ;
```

Attention :

```
int n = rand() ;
double t[n]; /*IMPOSSIBLE*/
```

Pointeurs :

```
int n = rand() ;
double *t = malloc(sizeof(double)*n) ;
double *u = t ;
free(t) ;
```

Pointeurs vs tableaux : *accès*

= *similitudes/équivalence*

```
#define N 10 ;
int n=rand()+N, i=rand()%N ;
double *t = malloc(sizeof(double)*n) ;
double u[N] ;

t[i] = u[i] = 5 ;
*(t+i) = *(u+i) = 6 ;

free(t) ;
```

→ t et u sont des adresses, accès aux éléments par [] ou par * de façon équivalente.

Pointeurs vs tableaux : résumé

- Déclaration/allocation de mémoire très *différentes* :
  - Utiliser un tableau pour une allocation statique
    → *nombre de cas restreint*
  - Utiliser un pointeur et malloc pour une allocation dynamique
    → *toujours possible*
- Accès *identiques* : les pointeurs et les tableaux sont des adresses, accès possible aux éléments par [ ] ou par * dans les deux cas.

Aujourd'hui

Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image (type, patch)
* Doxygen : outil de documentation du code

Tableaux 2D : comment les manipuler ?

Problème : `int t[5][6];`

- Les tableaux 2D ne sont que des tableaux 1D déguisés.
- Il n'est pas toujours possible d'utiliser l'indexation 2D (`t[i][j]`), notamment si les tailles ne sont pas fixées de manière statique.

Retour sur les 3 techniques possibles :

- Déclarer et utiliser un tableau 2D : ex. `int t[5][6];`
- Utiliser un pointeur sur une zone (1D)
- Utiliser un pointeur sur pointeur

Technique 1 : déclarer un tableau 2D

```
#define D1 5
#define D2 6

void init(int u[D1][D1]){
 int i,j;
 for (i=0;i<D1;i++)
  for (j=0;j<D1;j++)
   u[i][j]=0; /*accès 2D*/
}
int main(void){
 int t1[D1][D1];
 int t2[D2][D2];
 init(t1);
 init(t2); /*IMPOSSIBLE*/
 return 0;
}
```

- Déclaration 2D
- Accès 2D
- Mais : restriction au cas statique.

Approprié si la taille du tableau est unique.

Technique 2 : pointeur sur zone 1D

```
#define D1 5
#define D2 6

void init(int *u, int D){
 int i,j;
 for (i=0;i<D;i++)
  for (j=0;j<D;j++)
   u[i*D+j]=0; /*accès 1D*/
}
int main(void){
 int *t1=malloc(sizeof(int)*D1*D1);
 int *t2=malloc(sizeof(int)*D2*D2);
 init(t1,D1);
 init(t2,D2);
 return 0;
}
```

- Déclaration 1D facile
- Pas de restriction
- Accès 1D délicat , pas d'indexation 2D (nécessite une conversion)

Approprié si la conversion ne rend pas le codage illisible

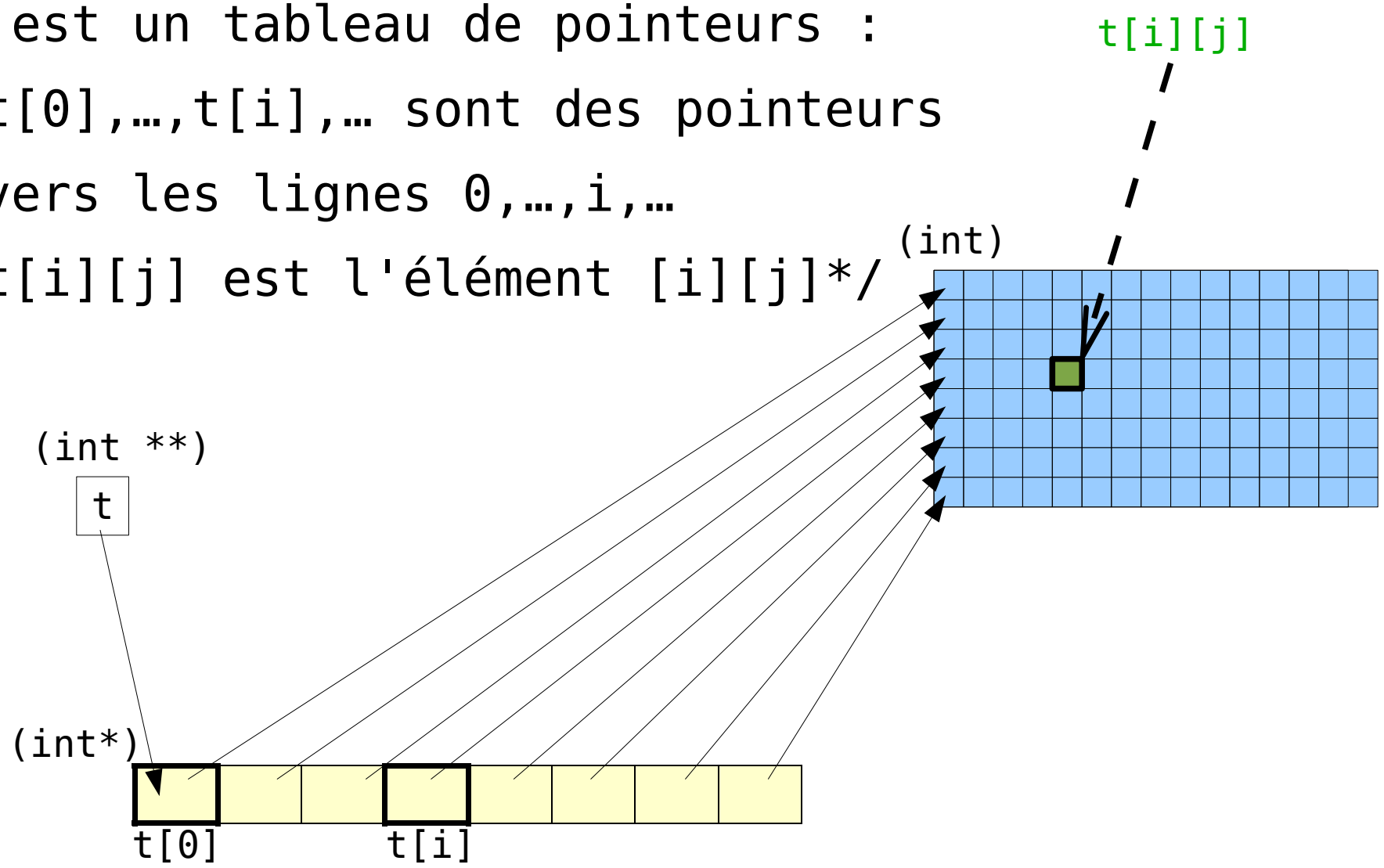
Peut-on avoir un accès 2D sans restriction ?

```
#define D1 5
#define D2 6

void init(int ...u..., int D){
 int i,j;
 for (i=0;i<D;i++)
  for (j=0;j<D;j++)
   u[i][j]=0; /*accès 2D*/
}
int main(void){
 ...t1=...;
 ...t2=...;
 init(t1,D1);
 init(t2,D2);
 return 0;
}
```

Technique 3 : pointeur sur pointeur

```
int **t ;
/* t est un tableau de pointeurs :
   - t[0],…,t[i],… sont des pointeurs
     vers les lignes 0,…,i,…
   - t[i][j] est l'élément [i][j]*/
```



Technique 3 : pointeur sur pointeur

```
#define D1 5
void init(int **u, int D){
 int i,j;
 for (i=0;i<D;i++)
  for (j=0;j<D;j++)
   u[i][j]=0; /*accès 2D*/
}
int main(void){
 int **t1=malloc(sizeof(int*)*D1);
 int **t2=malloc(sizeof(int*)*D2);
 int i;
 for (i=0;i<D1;i++)
  t1[i]=malloc(sizeof(int)*D1);
 init(t1,D1);
  /* idem pour t2 */
 return 0;
}
```

- Déclaration/initialisation plus difficile
- Pas de restriction
- Accès 2D (facile)

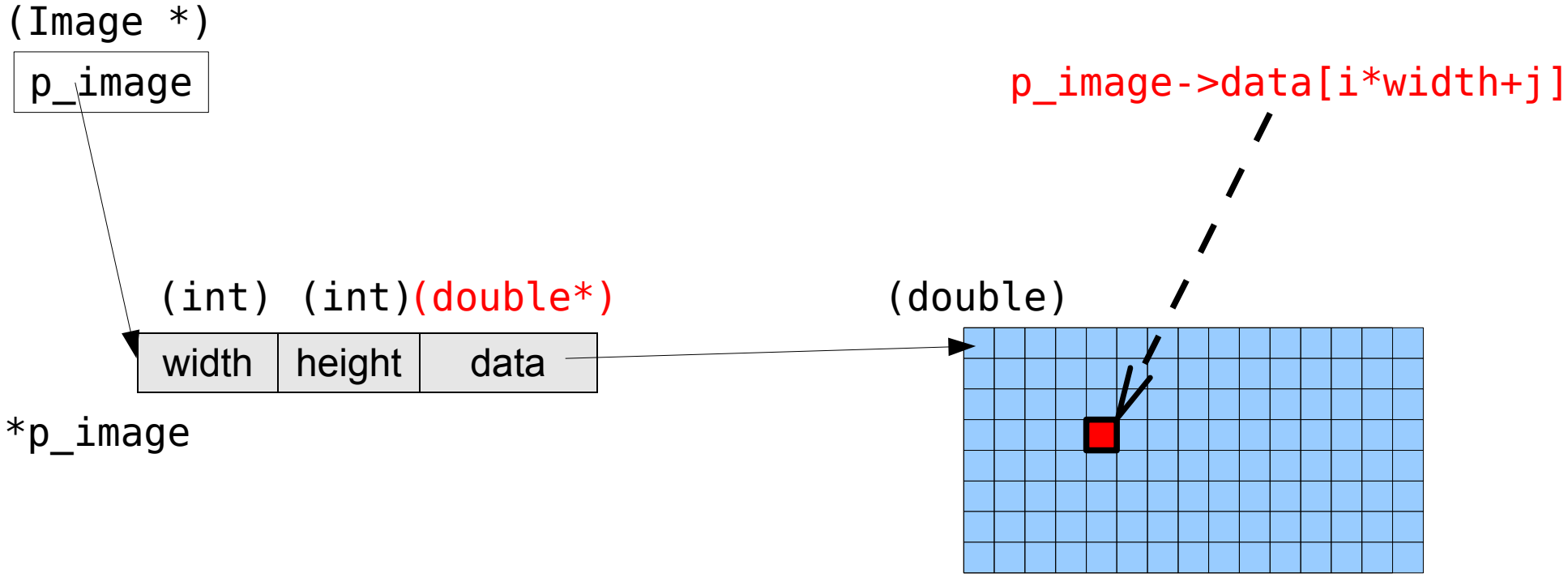
Souvent, il vaut mieux que la difficulté se situe à la création plutôt qu'à l'utilisation.

Aujourd'hui

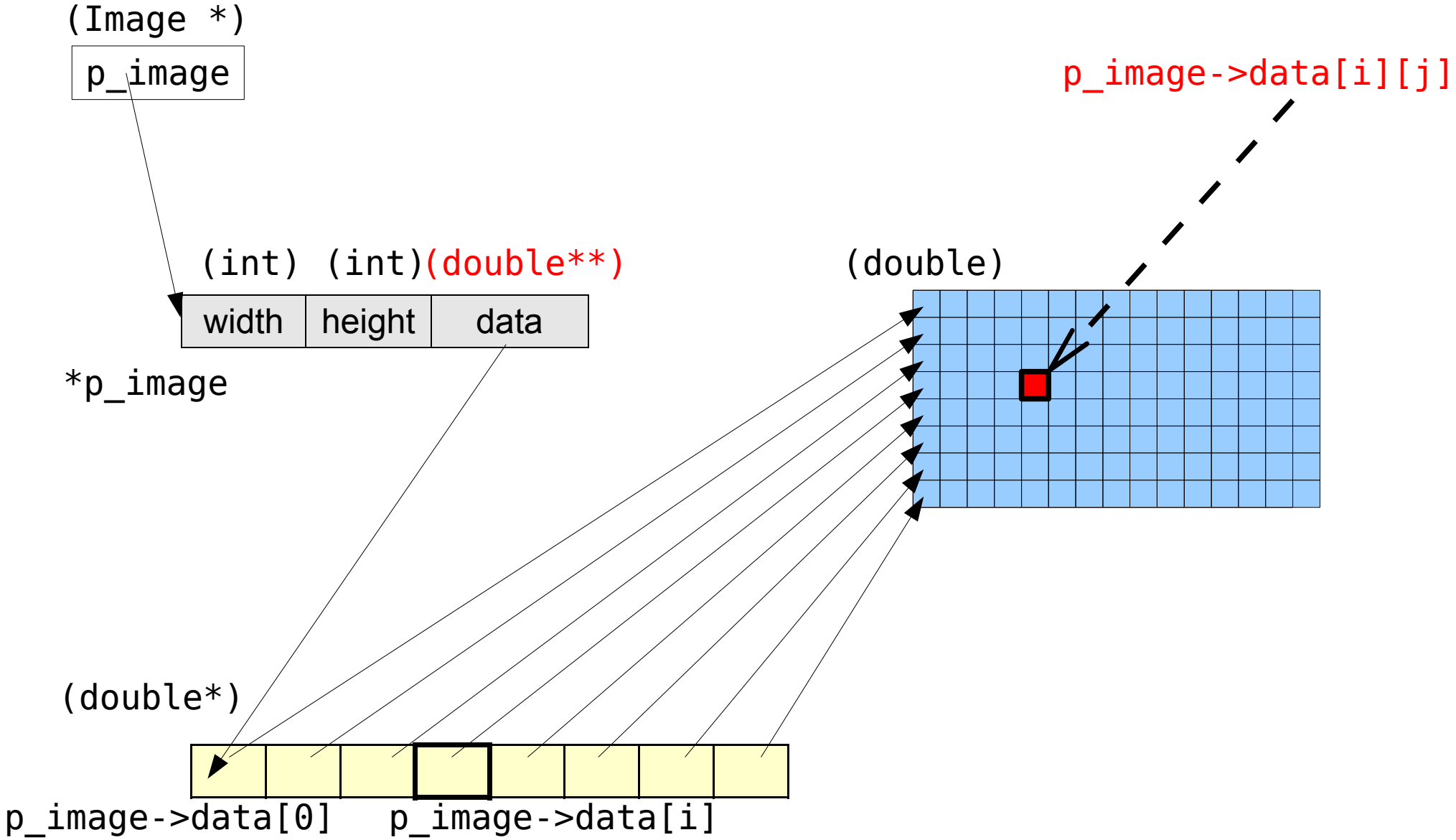
Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image
    * Nouveau type image
    * Traitement par patches
* Doxygen : outil de documentation du code

Type image avec accès 1D (TP 7)



Type image avec accès 2D (TP 8)



Aujourd'hui

Pratique du langage C (2)

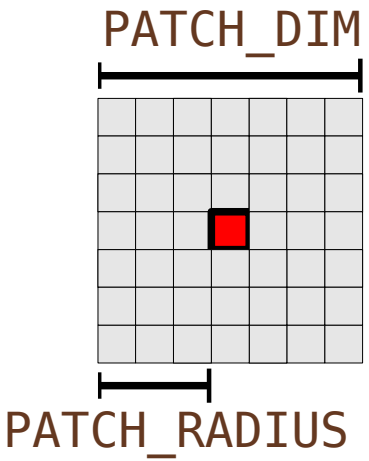
* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image
    * Nouveau type image
    * Traitement par patches
* Doxygen : outil de documentation du code

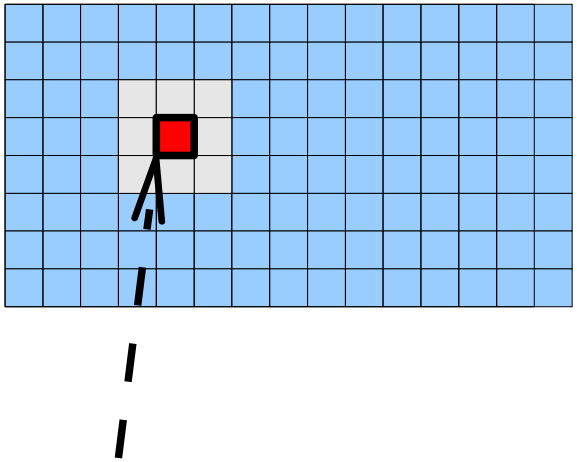
Notion de patch (voisinage d'un pixel)

- Patch = voisinage de taille impaire (typiquement 3x3)

- Exemple : patch 7x7

```
#define PATCH_DIM 7
#define PATCH_RADIUS (PATCH_DIM-1)/2
```



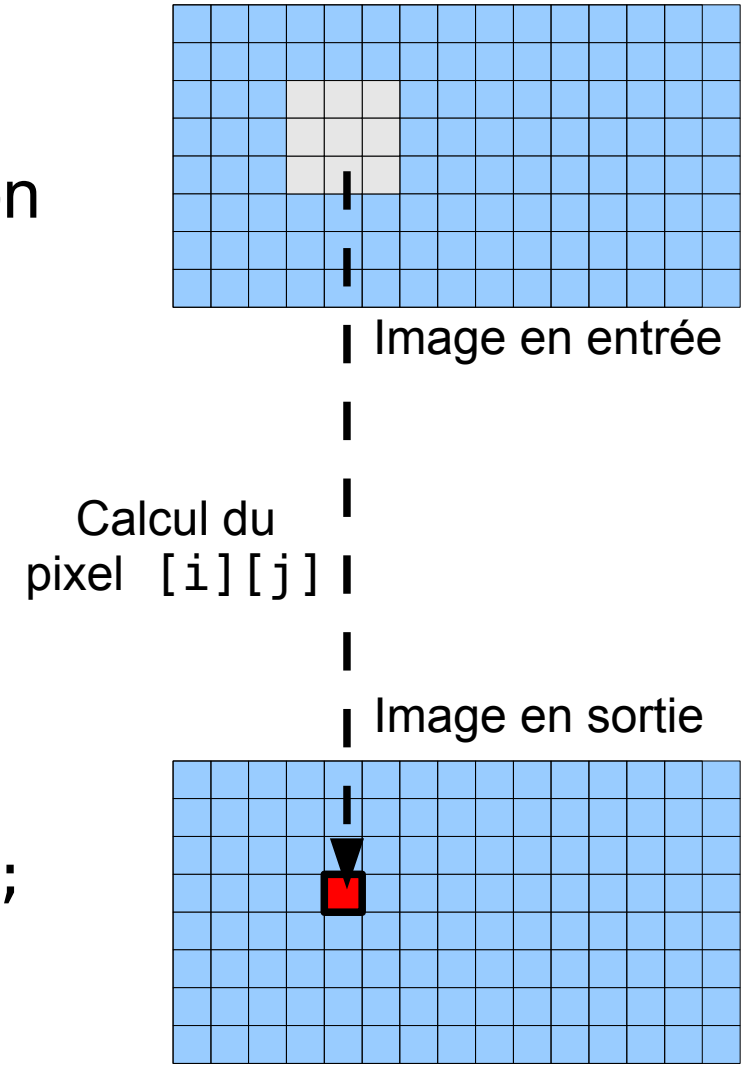


- Beaucoup de transformations se font à partir de patches

Notion de patch (voisinage d'un pixel)

- Principe :
  on crée le nouveau pixel à la position `[i][j]` en fonction du patch centré en `[i][j]`.
- Exemple : moyenne

```
p_image_res->data[i][j] =
( p_image->data[i-1][j-1]
+ p_image->data[i-1][j]
+ p_image->data[i-1][j+1]
+ …
+ p_image->data[i+1][j+1]) / 9 ;
```

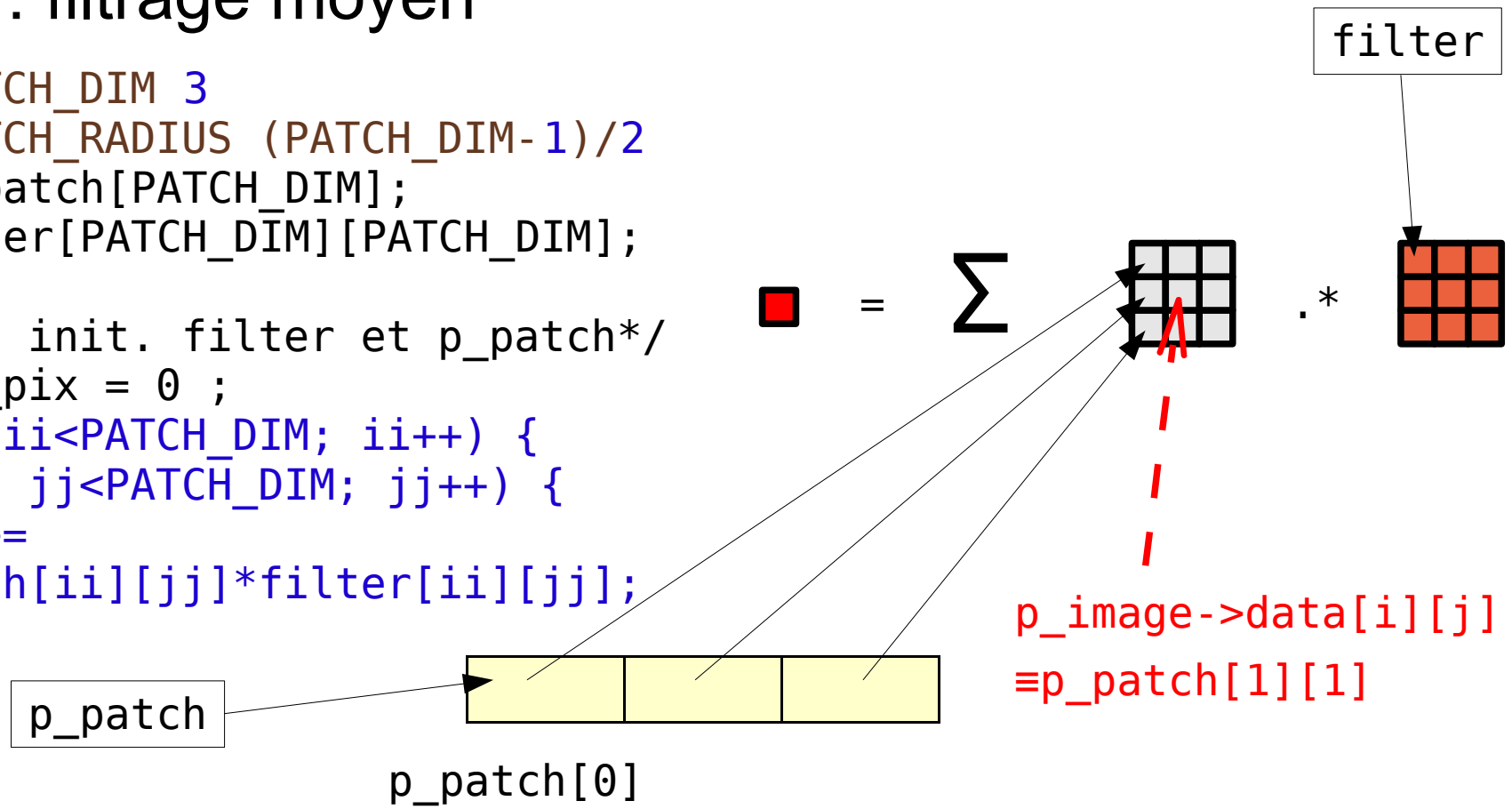


Notion de patch (voisinage d'un pixel)

Principe de mise en œuvre : se restreindre aux dimensions du patch

Exemple : filtrage moyen

```
#define PATCH_DIM 3
#define PATCH_RADIUS (PATCH_DIM-1)/2
double *p_patch[PATCH_DIM];
double filter[PATCH_DIM][PATCH_DIM];
int ii, jj;
/*A faire : init. filter et p_patch*/
double new_pix = 0 ;
for (ii=0; ii<PATCH_DIM; ii++) {
 for (jj=0; jj<PATCH_DIM; jj++) {
  new_pix +=
     p_patch[ii][jj]*filter[ii][jj];
 }
}
```



Aujourd'hui

Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image (type, patch)
* Doxygen : outil de documentation du code

Documenter son code pourquoi (et pourqui) ?

- Pour qu'un **utilisateur non-développeur** puisse utiliser le logiciel
- Pour qu'un **utilisateur développeur** puisse utiliser le code, comme bibliothèque par exemple.
- Pour qu'un **autre développeur** puisse créer une nouvelle version : corriger des erreurs, mettre à jour, ajouter des fonctionnalités, …
- Pour ne pas se perdre **soi-même** dans son propre code : par la suite, vous allez écrire typiquement des milliers de lignes de code pour le moindre projet.

Documentation pour l'utilisateur

- Comment installer (simplement) le logiciel ?
  Manuel d'installation (readme.txt, aide en ligne)
- Comment l'utiliser (simplement) ?
  Premiers pas (fichier pdf, aide en ligne)
- Description de toutes les fonctionnalités
  Manuel d'utilisation (fichier pdf, aide en ligne)

- Pas besoin de détails techniques et internes comme pour l'utilisateur développeur...

Documentation pour l'utilisateur-développeur

Par exemple pour une bibliothèque (idem pour programme)

- La documentation précédente
- La description haut-niveau du code source :
  - Quel langage ?
  - Quels services la bibliothèque propose-t-elle ? types, fonctions, etc. + dans quels fichiers
  - Pour chaque service, comment l'utiliser : types d'entrée et de sortie d'une fonction, comportement d'une fonction, …
- Pas besoin de détails encore plus fins comme pour le développeur du code...

Documentation pour le développeur

Par exemple pour une bibliothèque (idem pour programme)

- La documentation de l'utilisateur
- La documentation de l'utilisateur développeur
- Un commentaire interne au code

```
/* ceci est un commentaire interne */
```

  dès que c'est utile
  - Décrire les fonctions/types internes
  - Décrire le code pratiquement à chaque ligne

Le travail de documentation

- Essentiel : pour débuguer, pour diffuser
  Corolaire : un code non-documenté NE SERVIRA PAS
- Chronophage :
  - Plusieurs niveaux/cibles
  - Volume élevé de documentation
  - Plusieurs types de fichiers (sources, pdf, html, etc.)
  - Maintenance simultanée documentation+code

Doxygen : facilite le travail de documentation en réduisant les aspects chronophages.

Présentation de Doxygen

- Quoi ? Un logiciel (libre, multiplateforme, largement utilisé).
- Où ? http://www.stack.nl/~dimitri/doxygen/
- Pour quoi faire ? Générer automatiquement la documentation d'un code.
- Pour quels langages ? C, C++, C#, Objective-C, IDL, Java, VHDL, PHP, Python, Tcl, Fortran, D, …
- Principe :
  - Le développeur documente le code source seulement
  - Toute la documentation est générée automatiquement

2 étapes minimales

Il suffit de...

- Configurer le logiciel
- Avoir bien commenté son code

Prise en main rapide

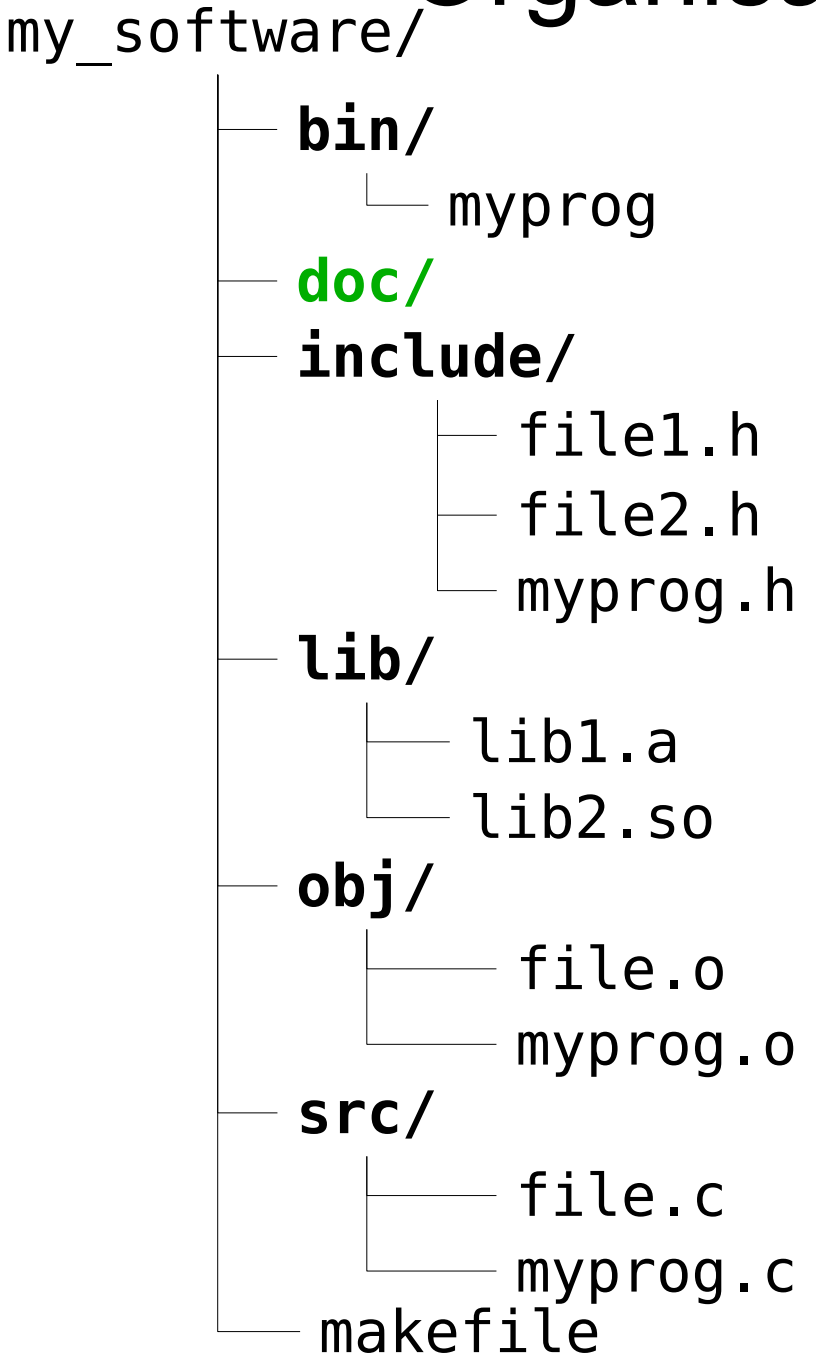
Doxygen nécessite juste un **fichier de configuration** pour indiquer comment générer la doc

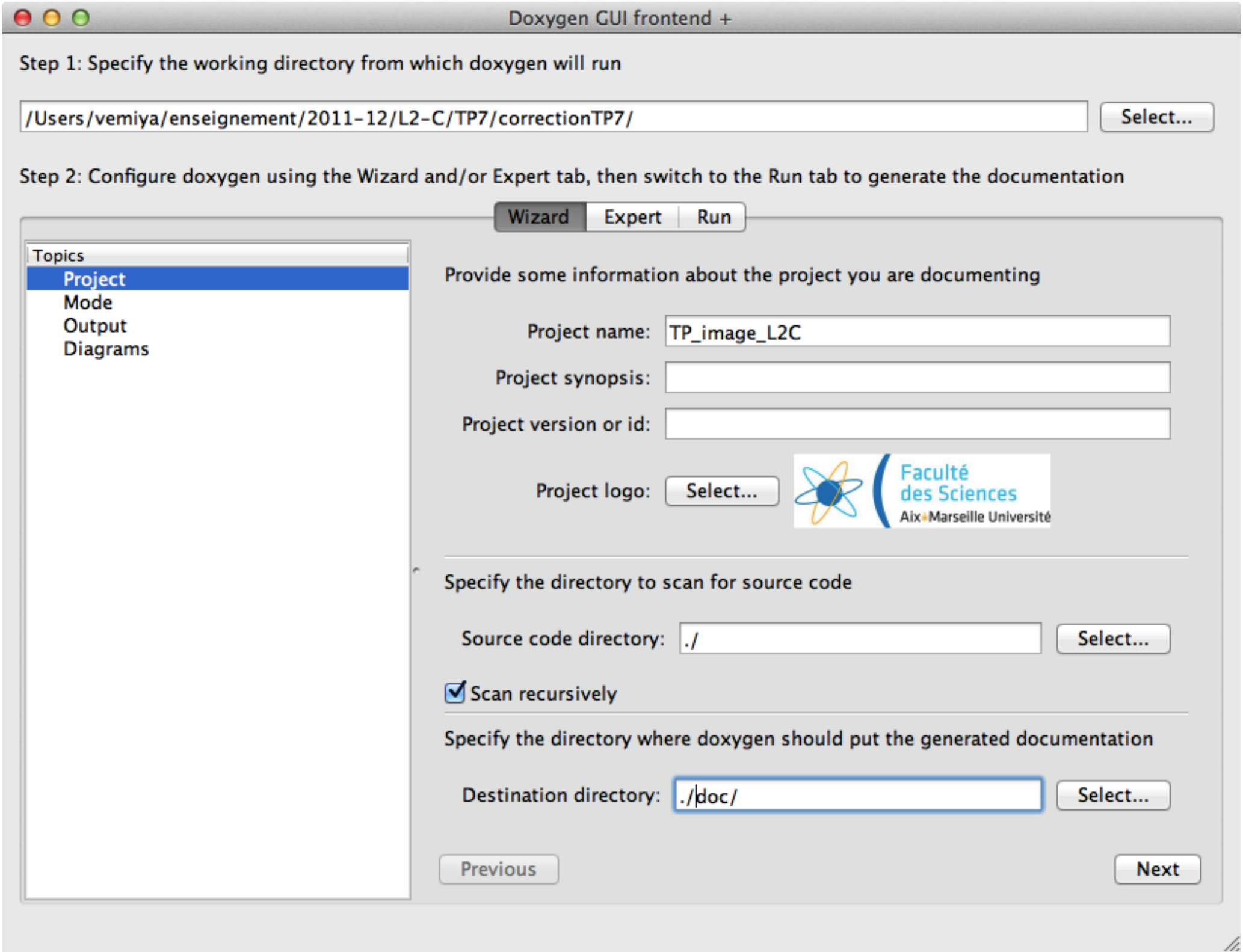
Bonne nouvelle :

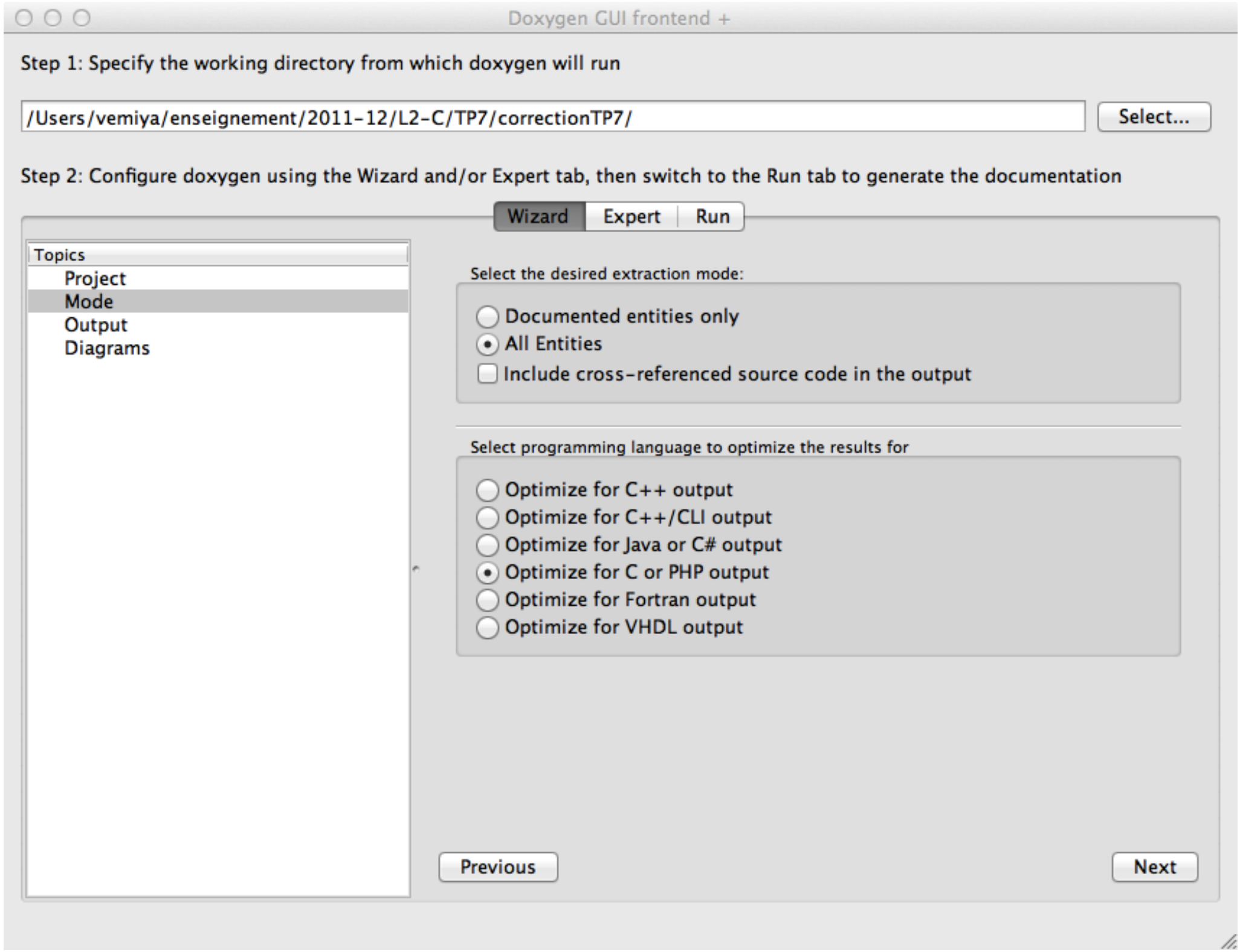
une interface graphique rend cela facilissime !

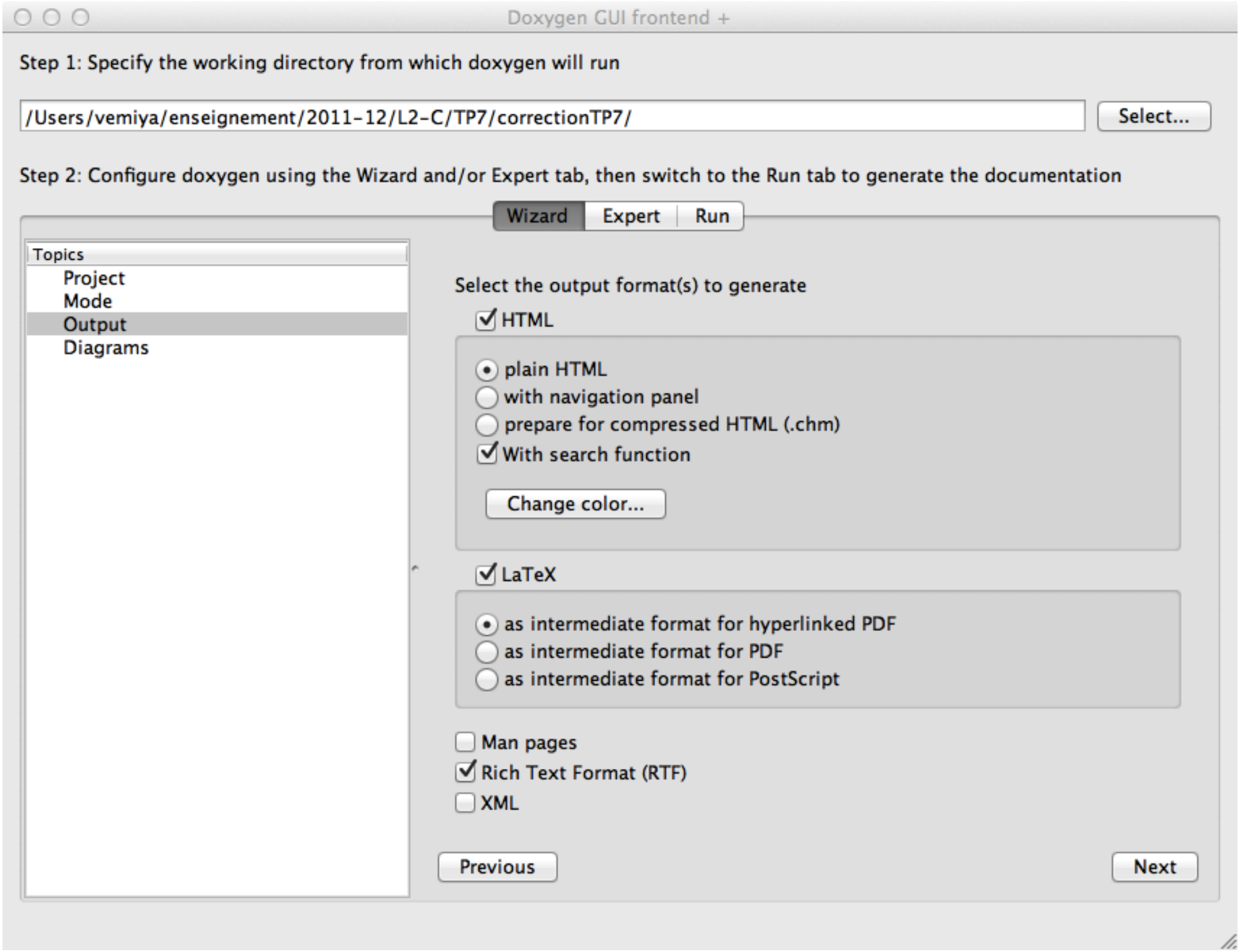
Exemple en 3 clics...

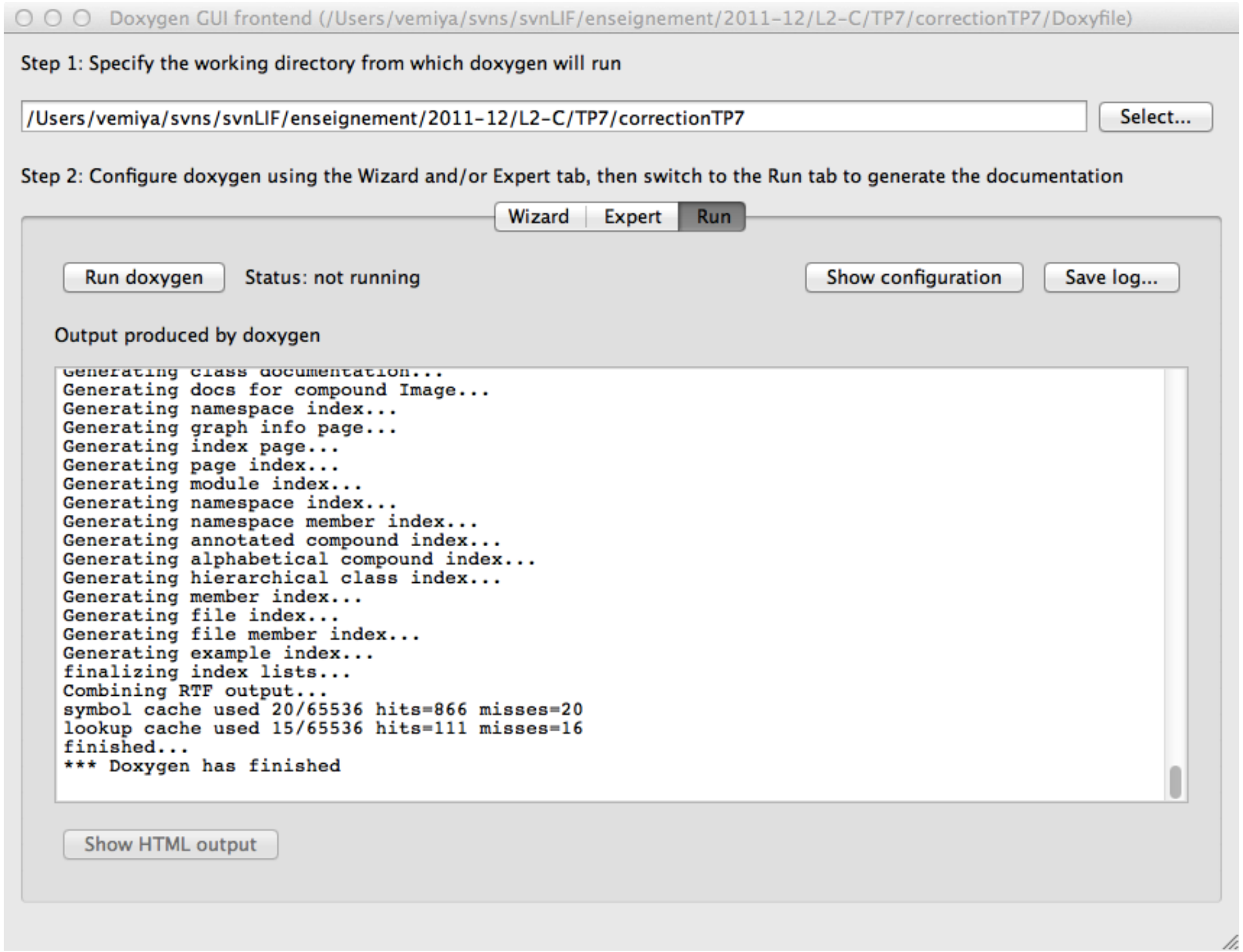
Organisation des fichiers




Doxygen GUI frontend +
Step 1: Specify the working directory from which doxygen will run
/Users/vemiya/enseignement/2011-12/L2-C/TP7/correctionTP7/
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Mode
Output
Diagrams
Provide some information about the project you are documenting
Project name:
TP_image_L2C
Project synopsis:
Project version or id:
Project logo:
Select...
Faculté
des Sciences
Aix-Marseille Université
Specify the directory to scan for source code
Source code directory:
./
Select...
Scan recursively
Specify the directory where doxygen should put the generated documentation
Destination directory:
./doc/
Select...
Previous
Next


Doxygen GUI frontend +
Step 1: Specify the working directory from which doxygen will run
/Users/vemiya/enseignement/2011-12/L2-C/TP7/correctionTP7/
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Mode
Output
Diagrams
Select the desired extraction mode:
Documented entities only
All Entities
Include cross-referenced source code in the output
Select programming language to optimize the results for
Optimize for C++ output
Optimize for C++/CLI output
Optimize for Java or C# output
Optimize for C or PHP output
Optimize for Fortran output
Optimize for VHDL output
Previous
Next


Doxygen GUI frontend +
Step 1: Specify the working directory from which doxygen will run
/Users/vemiya/enseignement/2011-12/L2-C/TP7/correctionTP7/
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Topics
Project
Mode
Output
Diagrams
Select the output format(s) to generate
HTML
plain HTML
with navigation panel
prepare for compressed HTML (.chm)
With search function
Change color...
LaTeX
as intermediate format for hyperlinked PDF
as intermediate format for PDF
as intermediate format for PostScript
Man pages
Rich Text Format (RTF)
XML
Previous
Next


Doxygen GUI frontend (/Users/vemiya/svns/svnLIF/enseignement/2011-12/L2-C/TP7/correctionTP7/Doxyfile)
Step 1: Specify the working directory from which doxygen will run
/Users/vemiya/svns/svnLIF/enseignement/2011-12/L2-C/TP7/correctionTP7
Select...
Step 2: Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation
Wizard
Expert
Run
Run doxygen
Status: not running
Show configuration
Save log...
Output produced by doxygen
Generating class documentation...
Generating docs for compound Image...
Generating namespace index...
Generating graph info page...
Generating index page...
Generating page index...
Generating module index...
Generating namespace index...
Generating namespace member index...
Generating annotated compound index...
Generating alphabetical compound index...
Generating hierarchical class index...
Generating member index...
Generating file index...
Generating file member index...
Generating example index...
finalizing index lists...
Combining RTF output...
symbol cache used 20/65536 hits=866 misses=20
lookup cache used 15/65536 hits=111 misses=16
finished...
*** Doxygen has finished
Show HTML output

NB

Ne pas oublier de sauvegarder le fichier de configuration, par exemple dans le même répertoire que le makefile.

2 étapes minimales

Il suffit de...

- Configurer le logiciel
- Avoir bien commenté son code

Documenter son code / Doxygen

1 seule règle à retenir

Les commentaires du type

```
/** ceci est inclus dans la doc */
```

sont pris en compte ; les commentaires du type

```
/* ceci est dans le code, pas dans la doc */
```

n'apparaissent pas dans la doc, ils ne servent qu'aux développeurs du code.

C'est tout ?!

Les autres règles et les mots clés s'apprennent facilement au fur et à mesure :

- Par défaut, un commentaire décrit l'élément qui le suit.
- \brief *description* : description synthétique suivie d'un saut de ligne et de la description détaillée
- \file *fichier* : description du fichier courant
- fonctions :
  - \param *paramètre* : description d'un paramètre d'entrée
  - \return : description de la sortie
- etc.

Aujourd'hui

Pratique du langage C (2)

* Retour sur les pointeurs et les tableaux
  * Rappels
  * Tableaux 2D : les (3) différentes techniques
  * Application au traitement d'image (type, patch)
* Doxygen : outil de documentation du code