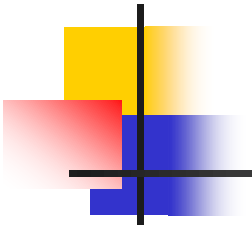
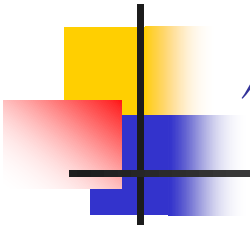


目录

- 使用Doxygen的目的.
- 安装Doxygen所需工具.
- 配置Doxygen
- Doxygen注释风格



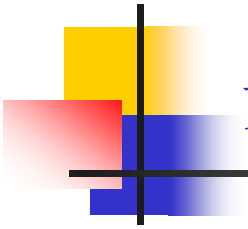
使用Doxygen的目的

- 生成模块文档，方便以后维护模块代码。
- 提高代码可读性。
- 提高项目代码的管理。
- 省略了自己写readme文件（或文档）。

安装Doxygen所需工具

- Doxygen(版本:1.5.2)
- Graphviz(版本: 2.12)
- iconv (GNU libiconv 1.9)
- fr(版本: 2.1.1.120)
- HTML Help Workshop (版本:4.74.8702.0)

(注:这些工具放在 \\dataserver\开发二部\开发二部_公共盘\白盒测试相关\生成文档工具)



安装Doxygen工具

1.先解压doxygen.rar;

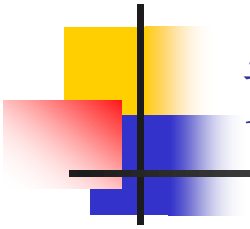
2.双击运行 doxygen-1.5.2-setup.exe;

3.安装软件提示语操作即可;

4.把create_chm.bat、html_foot拷贝到Doxygen安装目录的bin文件夹里.

5. 把 Doxygen安装目录的bin路径放在系统环境变量里;

6. Doxygen安装完成;



安装Graphviz工具

1. 先解压graphviz.rar;
2. 双击运行graphviz-2.12.exe;
3. 安装软件提示语操作即可;
4. Graphviz安装完成;

安装iconv、 fr工具

1.先解压iconv.rar和fr.rar;

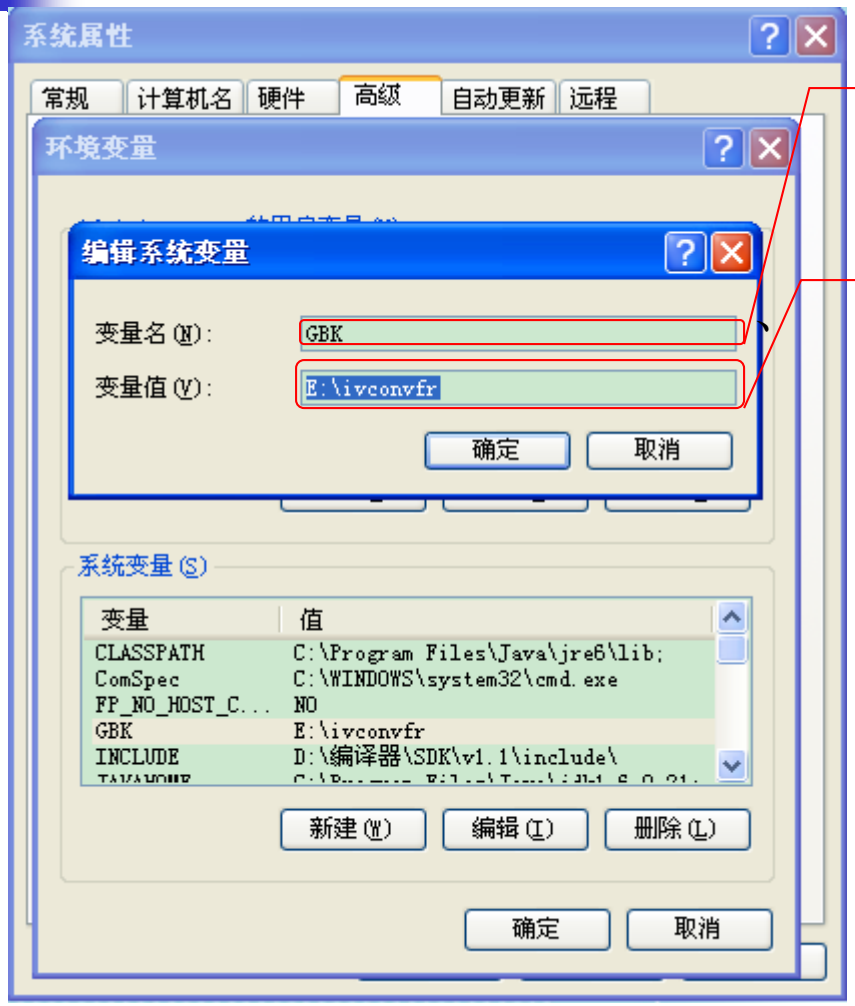
2.将iconv.rar和fr.rar分别解压出来的cygiconv-2.dll、cygintl-2.dll、cygwin1.dll、iconv.exe和fr.exe拷到“C:\WINDOWS\system32”;

3.安装完成;

注:上面安装能够运行成功,但是出现一个问题.就是我们编译下载环境模块时iconv的cygwin1.dll文件名明与cygwin的cygwin1.dll文件名相同导致的.所以编译不通过.

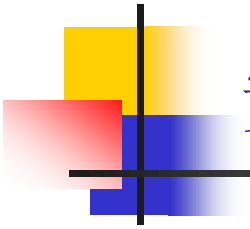
解决方法:在下页.

安装iconv、 fr工具解决方案



新建一个系统变量名为”GBK”,这我固定了.所以都以这个名字.

这个就是要放置iconv、 fr工具里文件的路径.(这个路径随自己编写)



安装HTML Help Workshop工具

1. 双击运行 htmlhelp.exe;
2. 安装软件提示语操作即可;
3. HTML Help Workshop安装完成;

注： HTML Help Workshop工具要安装在这个路径“X:\Program Files\HTML Help Workshop”（X：自己指定）.

附带操作

为了方便运行Doxygen工具与管理.为每个模块创建一文件夹(如：“TEST”),在TEST文件夹里再创建src、doc文件夹。Src文件夹存放源文件，doc文件夹存放Doxygen输出文件。顺便把doxygenWD.bat和Doxygen配置文件“Doxyfile”拷贝到doc文件夹下，如果要生成文档就双击doxygenWD.bat即可。（模块的目录结构如下 ）

注:Doxygen不支持中问路径,不要创建中文路径.

```
TEST
    |---- src
    |      |
    |      |---- test.c
    |      |---- test.h
    |
    |---- doc
    |      |
    |      |---- Doxyfile
    |      |---- doxygenWD.bat
```

配置Doxygen

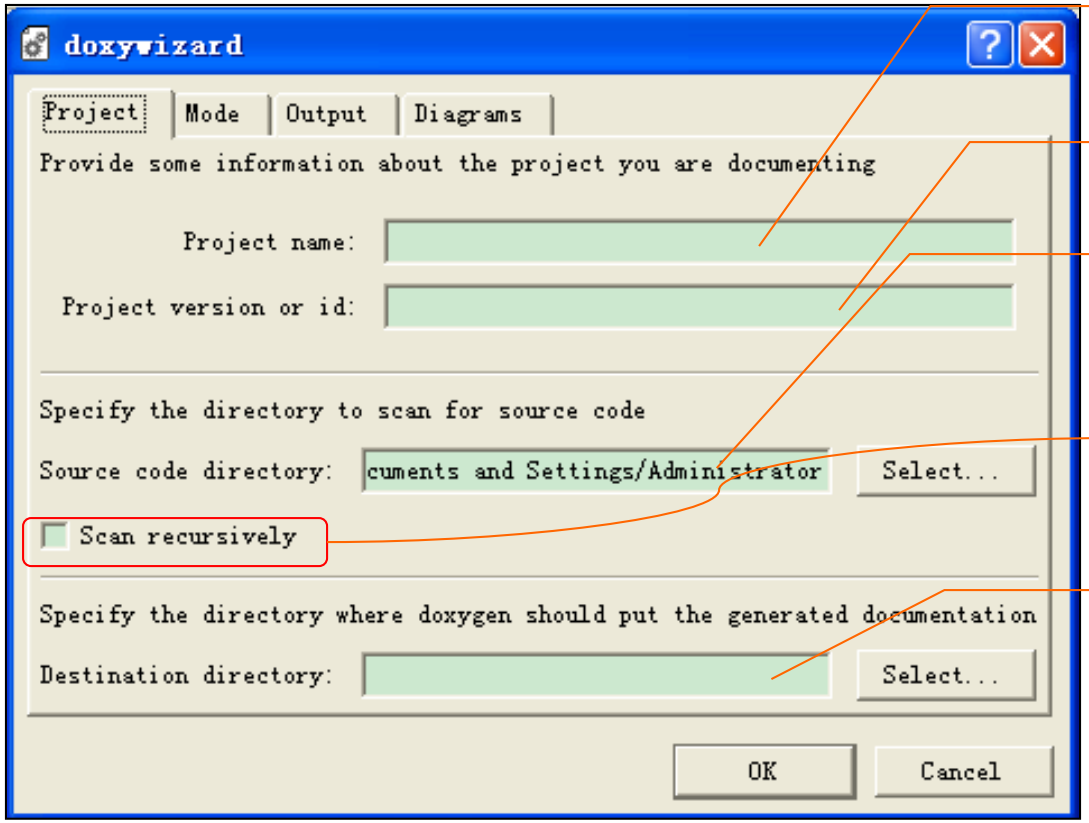
- 分为两种:

①向导(Wizard)模式：将快速配置大多数重要设定,其他选项保持默认值。

②专家(Expert)模式：进入全部的配置选项。

向导(Wizard)模式

- 向导(Wizard)对话框----Project相关选项



项目名称，将作为于所生成的程序文档首页标题.

文档版本号，可对应于项目版本号 .

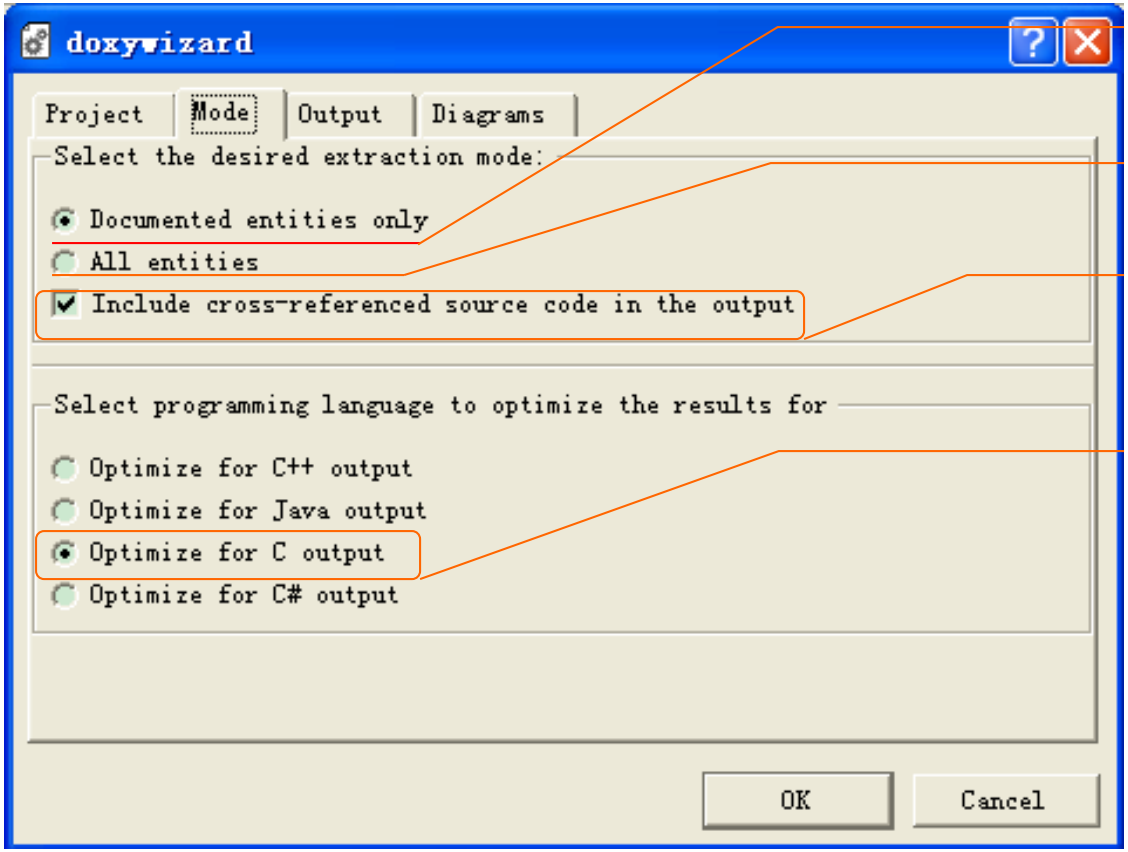
存放要生成项目文档的项目源文件路径.

如果选上，就会连子文件夹的文件一起生成文当。没选，就只会生成本文件夹的文件文档.

这里填入生成文档文件输出路径,这里只填文件夹名字就可以了.为了大家工作方便,就规定统一写output文件夹

向导(Wizard)模式

- 向导(Wizard)对话框----Mode相关选项



选择这个,只生成文档实体,不会生成源文件文档.

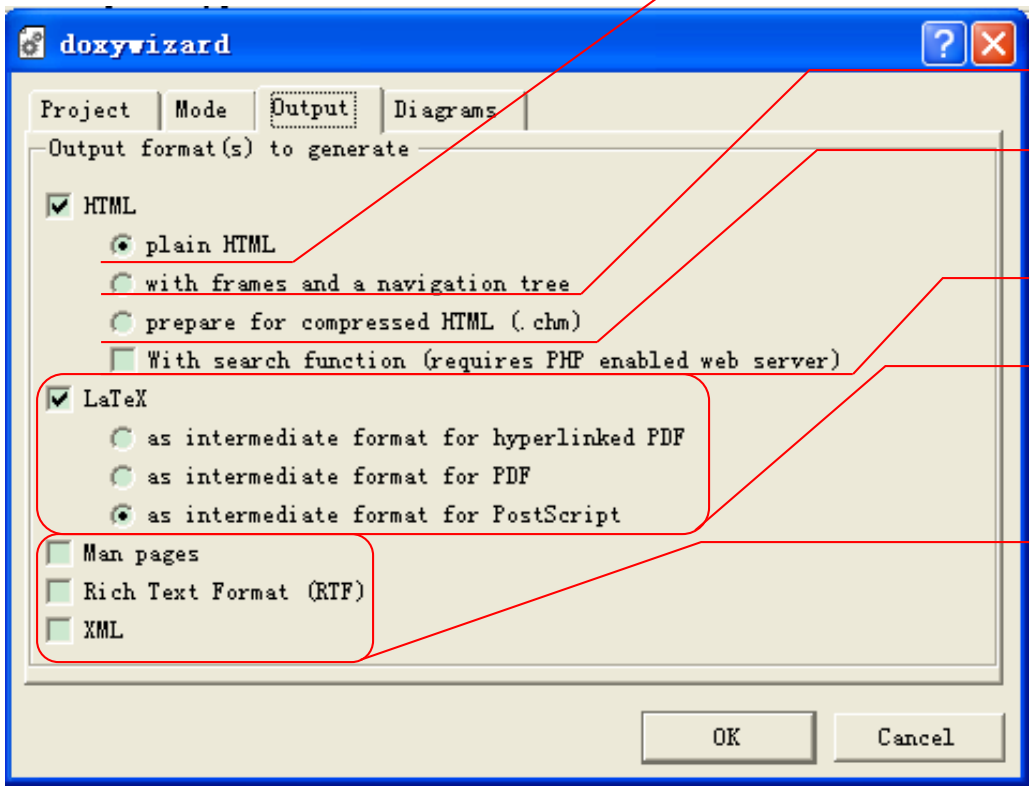
选择这个,会生成全部实体.因此,要选择这个.

选择这个,把源文件内容导入到项目文档里.(建议选择)

选择项目文档输出格式,测试C语言,就选择个.

向导(Wizard)模式

- 向导(Wizard)对话框----Output相关选项



生成普通模式的HTML

生成文件列表格的HTML

生成chm文件格式的HTML,因此,就选这个.

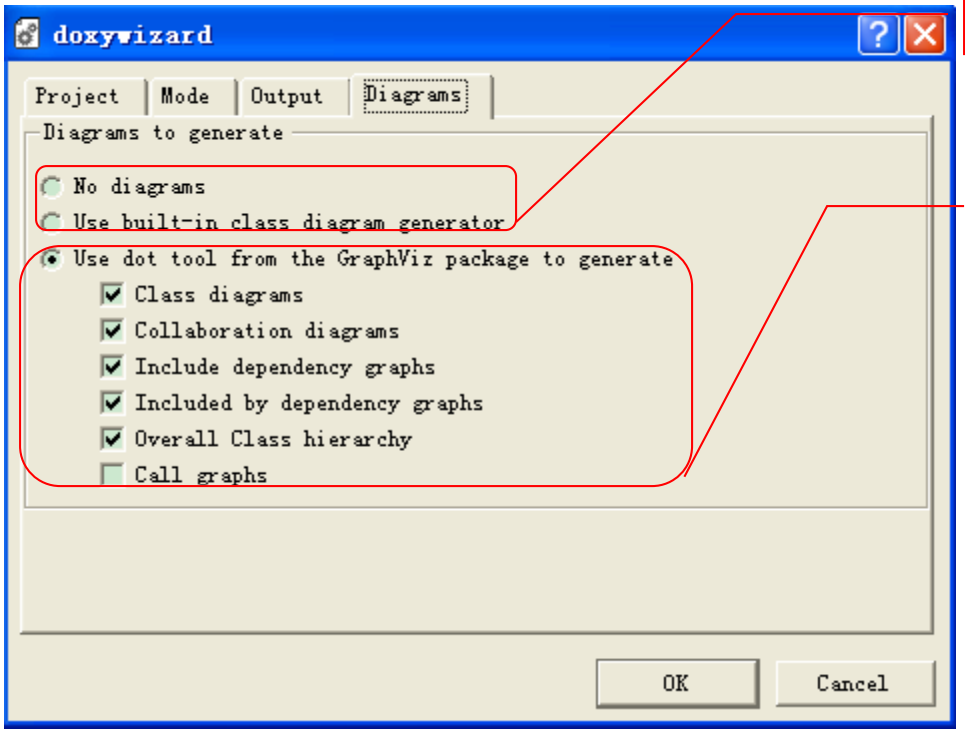
附带查找功能,一般不选择.

这项没用到,因此把它选择取消.

这三项不要选择

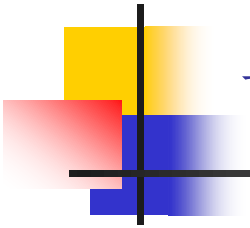
向导(Wizard)模式

- 向导(Wizard)对话框----Diagrams相关选项



这两个不选择.

全选择.通过这个GraphViz工具生成图表:
(1).类的图表.
(2).协作图表.
(3).包含文件图表(调用图表).
(4).被包含文件图表(被调用图表).
(5).整个类层次结构图表.
(6).调用图表(函数,文件).



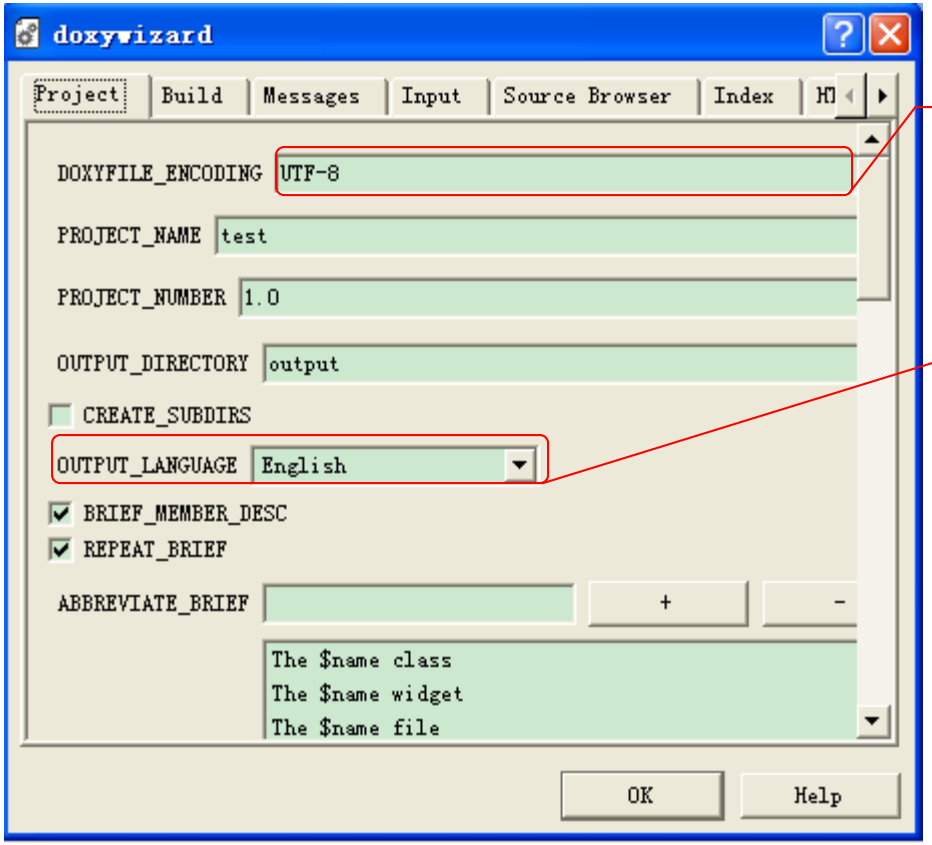
专家(Expert)模式

由于经过向导(Wizard)模式快速的配置，所以专家(Expert)模式大部分也相应配置好了。

那么，下面主要配置细节问题，其他就不详细介绍都保持默认选择就可以。

专家(Expert)模式

- 专家(Expert)对话框----Project相关选项

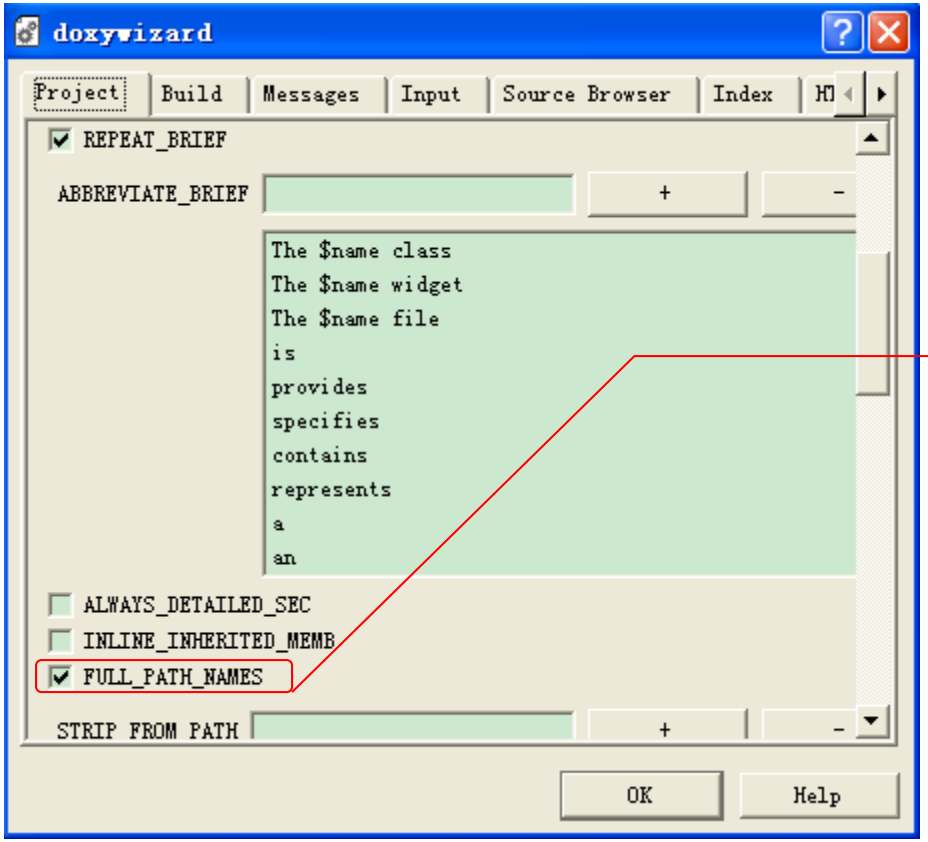


DOXYFILE_ENCODING是Doxyfile的文本编码。如果文件中有中文字符，可以填写GBK。把UTF-8改为GBK。

输出语言相当于程序资源，选择Chinese。

专家(Expert)模式

- 专家(Expert)对话框----Project相关选项

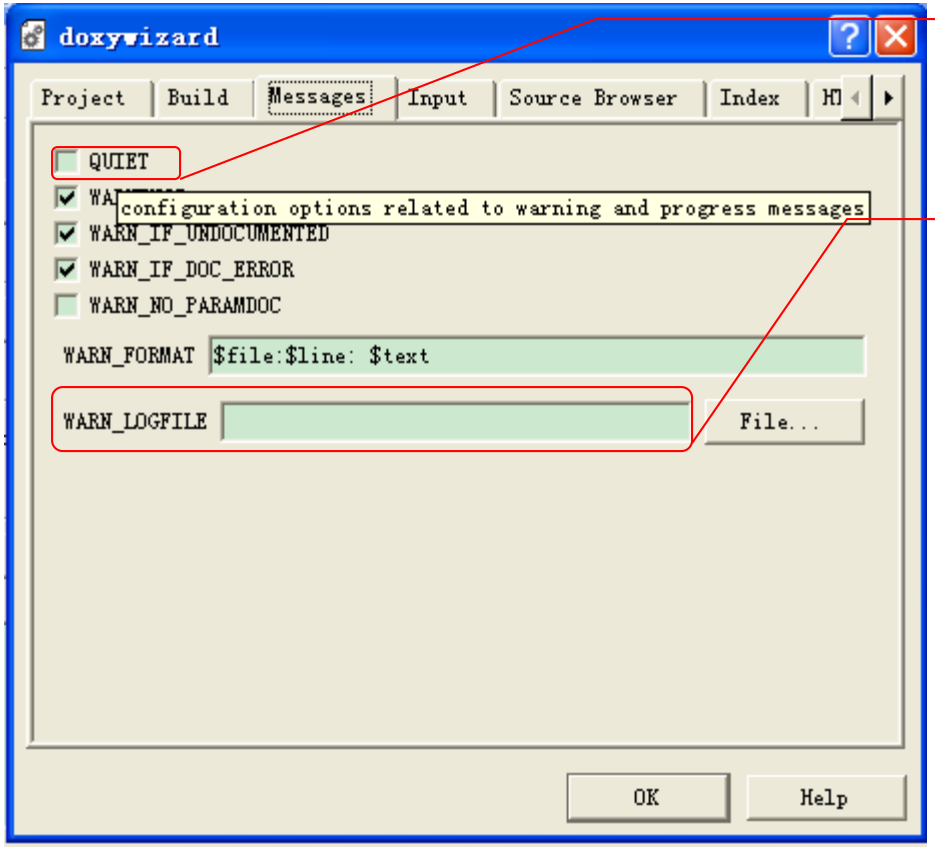


取消选项,不然会显示全路径:如图1.

图1

专家(Expert)模式

- 专家(Expert)对话框----Messages相关选项

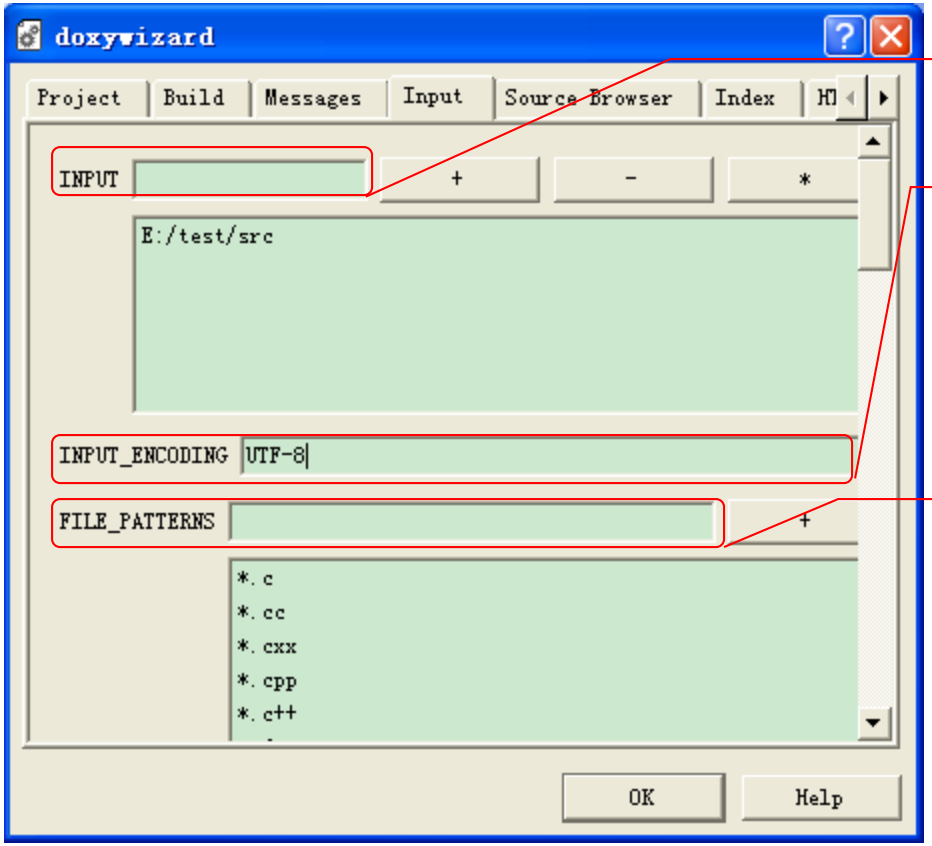


让 doxygen 静悄悄地为你生成文档，只有出现警告或错误时，才在终端输出提示信息(不选择).

将WARN_LOGFILE填写为error.txt。这样，Doxygen会将编译时出现的警告和错误保存在error.txt ，这样可以对照修改。

专家(Expert)模式

- 专家(Expert)对话框----Input相关选项



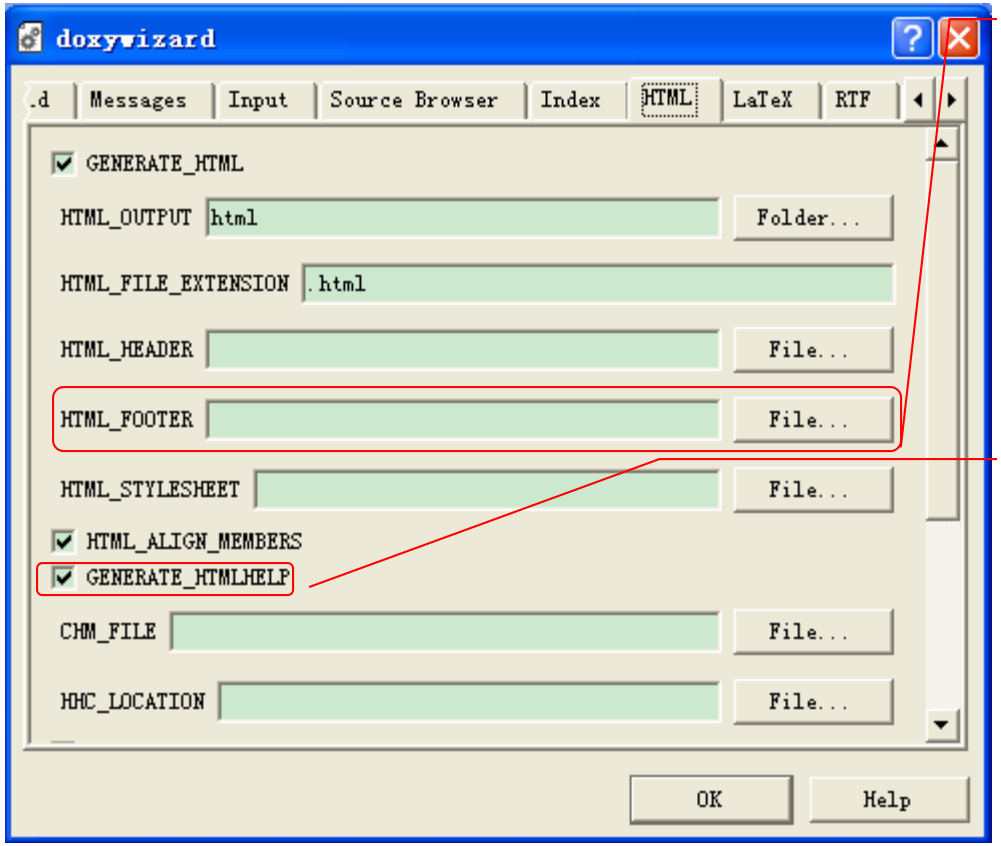
指定输入源文件目录（INPUT）.

输入文件编码（INPUT_ENCODING）改为GBK.

对文件过滤选项,我们这里只填*.c *.h.

专家(Expert)模式

- 专家(Expert)对话框----Messages相关选项



可以通过参数HTML_HEADER和HTML_FOOTER定制页面，参数值是包含定制内容的文件名. 这里我填了html_foot 文件,内可以打开html_foot 文件看. 这里只看效果图:如图2.

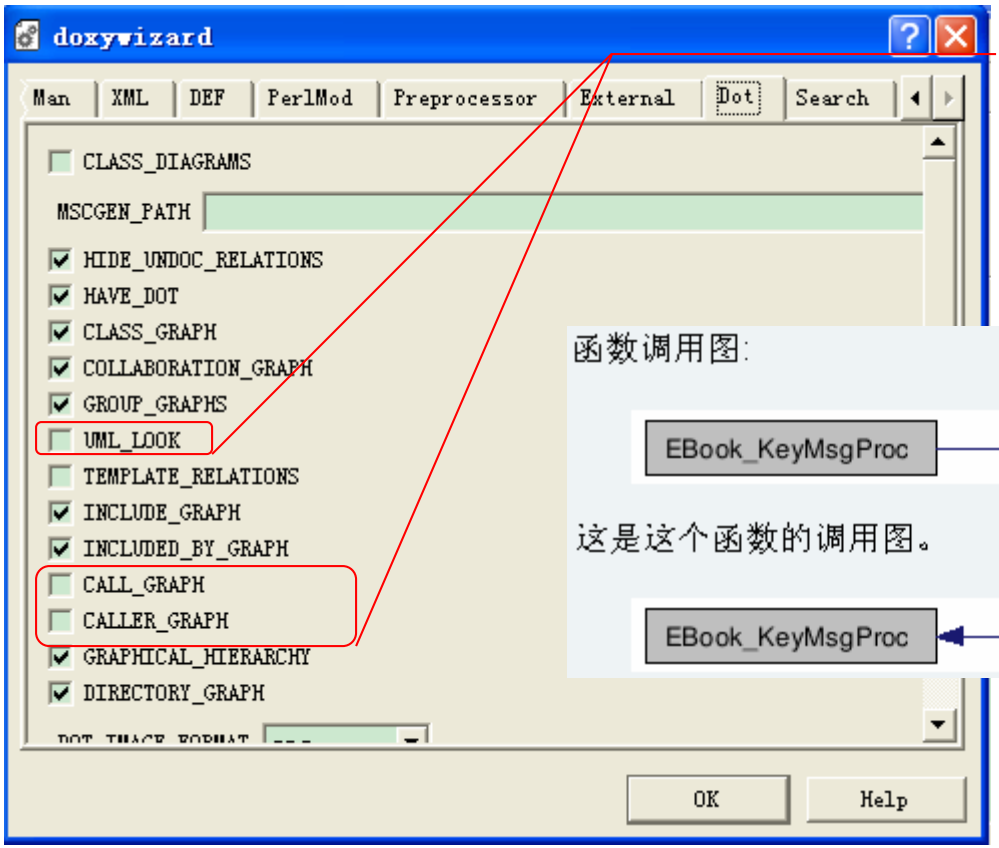
步步高第二事业部Doxygen工具获取

图2

选择GENERATE_HTMLHELP后，Doxygen会准备生成chm文件需要的项目文件、目录文件和索引文件。

专家(Expert)模式

- 专家(Expert)对话框----Dot相关选项



可以选上UML_LOOK、CALL_GRAPH和CALLER_GRAPH。CALL_GRAPH是本函数调用其它函数的示意图.效果如:图3.

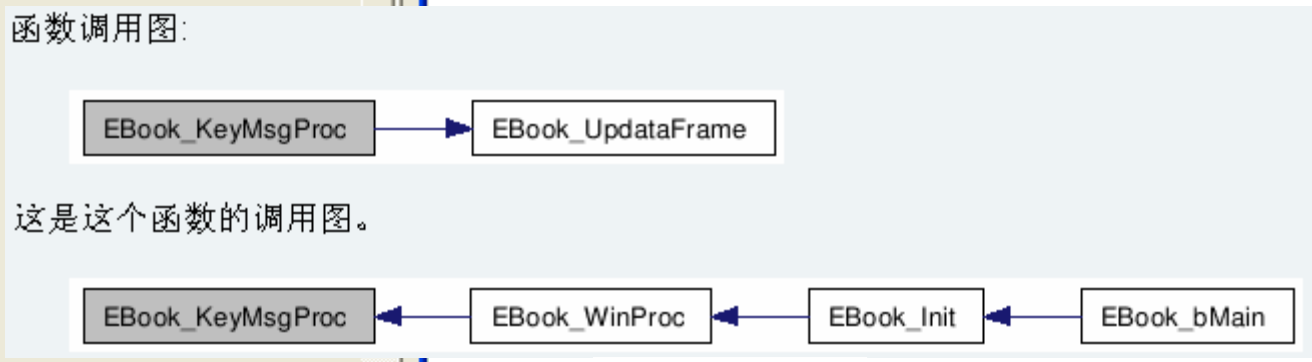
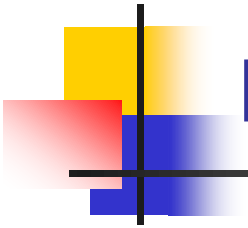


图3



Doxygen注释风格

Doxygen指令目的为了生成更丰富与可读性更强的文档。所以总结5类常用的注释风格说明。

- 变量、宏定义、类型定义。
- 枚举类型定义、结构体类型定义类似。
- 函数定义。
- 模块定义（单独显示一页 ）。
- 分组定义（在一页内分组显示）。

变量、宏定义、类型定义简要说明

变量、宏定义、类型定义注释风格类似。

格式：

```
/** 简要说明文字 */
变量（宏定义或类型定义）
```

/** …*/这是固定格式，还要注意/**这2个“**”不能少也不能多。其他注释风格也是这样的。

如：

```
/** 简要说明文字 */
#define FLOAT float

/** @brief 简要说明文字（在前面加 @brief 是标准格式） */
#define MIN_UINT 0

/**
 * 分行的简要说明 \n
 * 这是第二行的简要说明
 */
int b;
```

“@brief ”是注释指令，“@”也可以用”\”.

“\n”作用是回车换行.

注：不文档化局部变量，只文档化全局变量。

枚举类型定义、结构体类型定义

枚举类型定义、结构体类型定义注释风格类似。

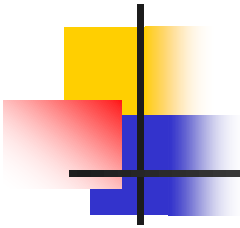
格式：

```
/** 简要说明文字 */
 typedef 类型 结构体名字
 {
    成员1, /**< 简要说明文字 */
    成员2, /**< 简要说明文字 */
    成员3, /**< 简要说明文字 */
 }结构体别名;
```

类型：enum ， struct

注意，要以 “<” 小于号开头，如过不用“<”，那这行的注释就会被Doxygen认为是成员2的注释。

注：所有的结构体类型定义要放在枚举类型定义前面。



函数定义

指令格式:

执行符指令操作符

执行符：@或\两个。

指令操作符：param、return、retval、note、 pre、 par、 code、endcode、see、deprecated。（其实还有很多，这里的指令操作符已经够我们函数用了，所以不再列举。）

特殊符号指令符：-、#、::。（这三符号对于我们也够用了）

Param 指令操作符讲解

Param：指定函数参数指令操作符。

Param 格式如下：

@param 参数名 参数说明

@param [in] 参数名 参数说明

@param [out] 参数名 参数说明

例：

```
/**
 *文件关闭函数
 * @param file文件编号。
 */
int CloseFile(int file);
```

Doxygen生成效果

return 指令操作符讲解

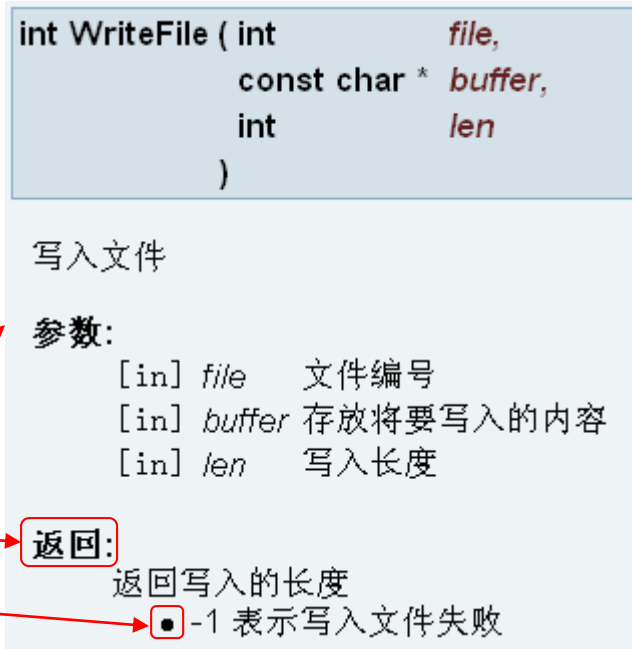
return：指定函数返回说明指令操作符。

return格式如下：

@return 简要说明

例:

```
/**
 * 写入文件
 * @Param [in]  file 文件编号
 * @Param [in] buffer 存放将要写入的内容
 * @Param [in]  len写入长度
 * @return 返回写入的长度
 * - -1 表示写入失败
 */
int WriteFile(int file, const char* buffer, int len);
```



retval指令操作符讲解

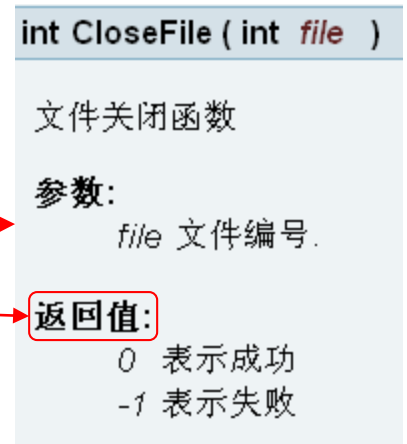
retval：指定函数返回值说明指令操作符。(注:更前面的return有点不同.这里是**返回值**说明)

retval格式如下：

@retval 返回值 简要说明

例:

```
/**
 *文件关闭函数
 * @param  file文件编号。
 * @retval  0  表示成功
 * @retval  -1 表示失败
 */
int CloseFile(int file);
```



note 指令操作符讲解

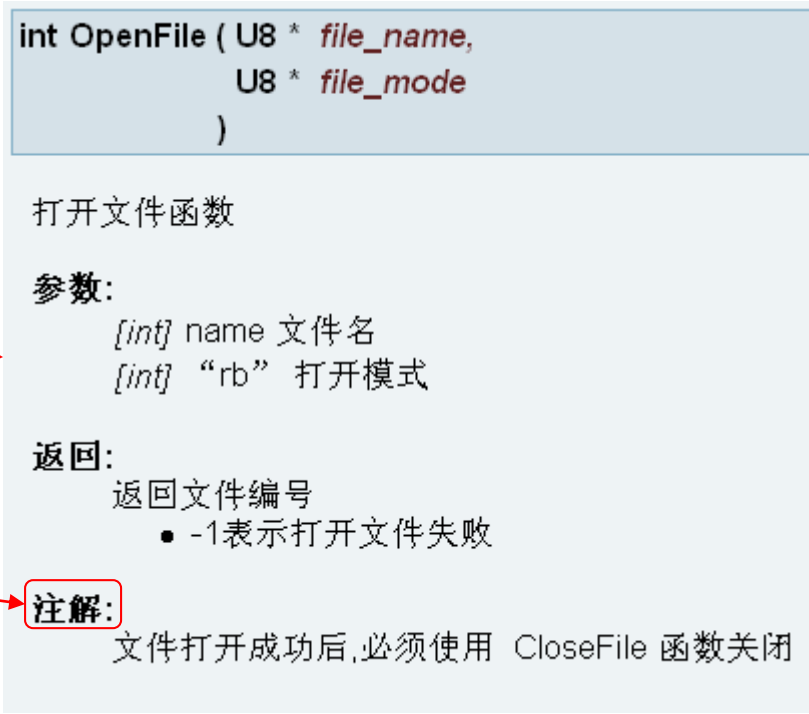
note：指定函数注意项事或重要的注解指令操作符。

note格式如下：

@note 简要说明

例：

```
/**
 *打开文件函数
 *@Param [in]  name  文件名
 *@Param [in]  “rb” 打开模式
 *@return 返回文件编号
 *- -1表示打开文件失败
 *@note 文件打开成功后,必须使用 CloseFile 函数关闭
 */
int OpenFile(U8* file_name, U8* file_mode);
```



pre 指令操作符讲解

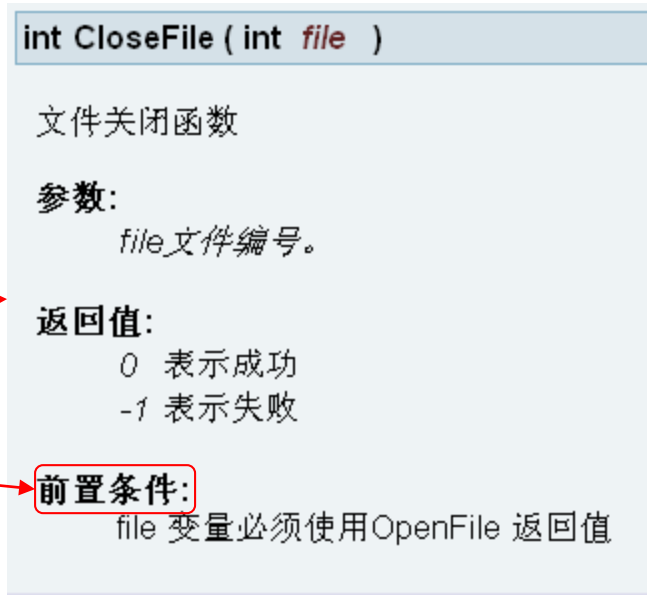
pre：指定函数前置条件指令操作符

pre格式如下：

@pre 简要说明

例：

```
/**
 *文件关闭函数
 * @param  file文件编号。
 * @retval  0  表示成功
 * @retval  -1 表示失败
 * @pre file 变量必须使用OpenFile 返回值
 */
 int CloseFile(int file);
```



par指令操作符讲解

par：指定扩展性说明指令操作符讲。（它一般跟code、endcode一起使用 ）

par格式如下：

@par 扩展名字

例:

```
/**
 * 打开文件函数
 * @Param [in]  name  文件名
 * @Param [in]  “rb” 打开模式
 * @return 返回文件编号
 * - -1表示打开文件失败
 * @par 示例:
 * @code
  //用文本只读方式打开文
  int f = OpenFile(”c:\\test.txt”, “rb”);
 * @endcode
 */
int OpenFile(U8* file_name, U8* file_mode);
```

```
int OpenFile ( U8 * file_name,
               U8 * file_mode
             )
```

打开文件函数

参数:

[in] name 文件名

[in] “rb” 打开模式

返回:

返回文件编号

- -1表示打开文件失败

示例:

```
//用文本只读方式打开文
int f = OpenFile(” c:\\test.txt” , “rb” );
```

code、endcode指令操作符讲解

code、endcode：指定

code、endcode格式如下：

@code

简要说明(内容)

@endcode

例:

```
/**
 * 打开文件函数
 * @Param [in]  name  文件名
 * @Param [in]  “rb” 打开模式
 * @return 返回文件编号
 * - -1表示打开文件失败
 * @par 示例:
 * @code
  //用文本只读方式打开文
  int f = OpenFile(”c:\\test.txt”, “rb”);
 * @endcode
 */
int OpenFile(U8* file_name, U8* file_mode);
```

```
int OpenFile ( U8 * file_name,
               U8 * file_mode
             )
```

打开文件函数

参数:

[in] name 文件名

[in] “rb” 打开模式

返回:

返回文件编号

- -1表示打开文件失败

示例:

```
//用文本只读方式打开文
int f = OpenFile(” c:\\test.txt” ,  “rb” );
```

see指令操作符讲解

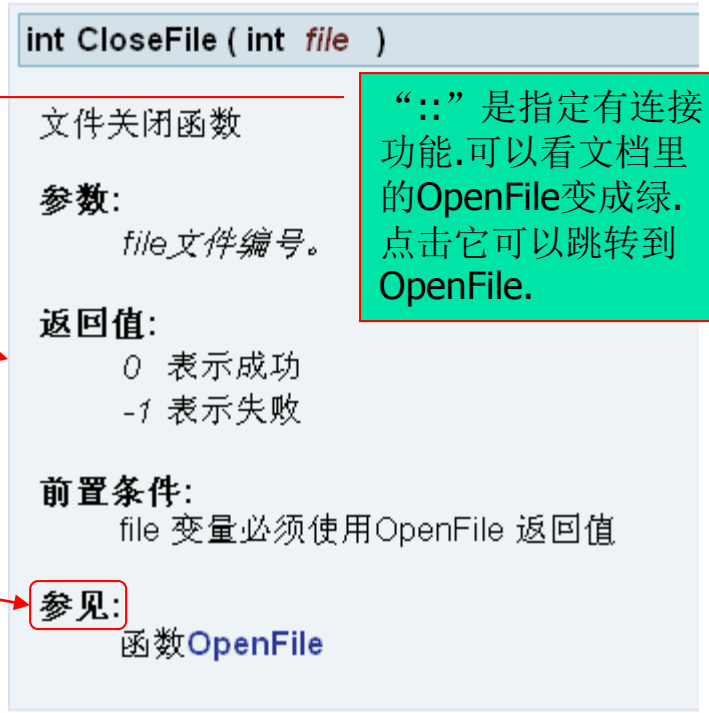
see：指定参考信息。

see格式如下：

@see 简要参考内容

例：

```
/**
 *文件关闭函数
 * @param  file文件编号。
 * @retval  0  表示成功
 * @retval  -1 表示失败
 * @pre file 变量必须使用OpenFile 返回值
 * @see 函数::OpenFile
 */
int CloseFile(int file);
```



deprecated指令操作符讲解

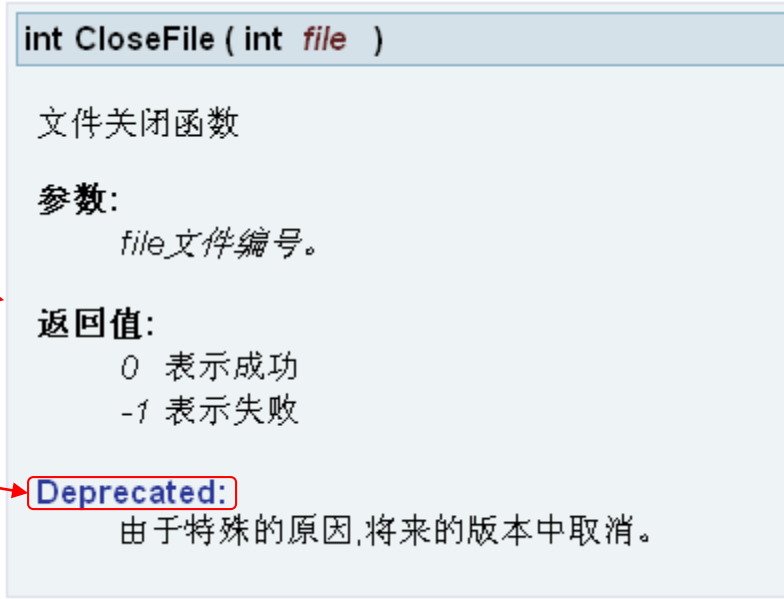
deprecated：指定函数过时指令操作符。

deprecated格式如下：

@deprecated 简要说明

例：

```
/**
 *文件关闭函数
 * @param  file文件编号。
 * @retval  0  表示成功
 * @retval  -1 表示失败
 * @deprecated 由于特殊的原因,将来的版本中取消。
 */
 int CloseFile(int file);
```



-、#、::指令操作符讲解

-：生成一个黑心圆.

-#：指定按顺序标记。

::：指定连接函数功能。（注：空格和“:”有连接功能,但建议还是使用”::”。只对函数有用。）

它们格式如下: (-和::例子前面有了,就介绍-#例子。)

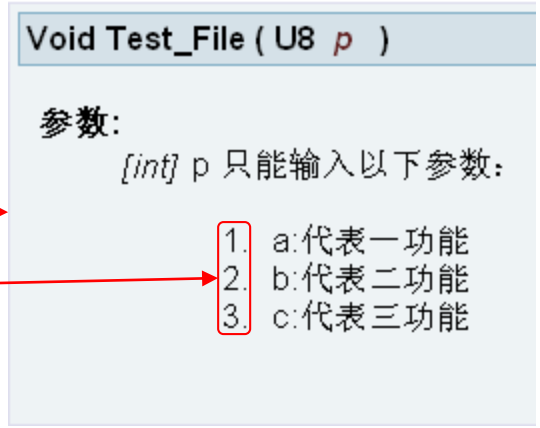
- 简要说明

-# 简要说明

::函数名

例：

```
/**
 * @param [in] p 只能输入以下参数：
 * -# a:代表一功能
 * -# b:代表二功能
 * -# c:代表三功能
*/
Void Test_File(U8 p);
```



模块定义（单独显示一页 ）

模块定义格式:

```
/**
 * @defgroup 模块名  页的标题名
 * @{
 */
 … 定义的内容 …
/** @} */
```

模块名只能英文,这个可以随便取.在一个源文件里不能相同.

跟c语言{一样起作用域功能.

例：

```
/**
  * @defgroup aa ebookmain.c
  * @{
  */
     … 定义的内容 …
  /** @} */
```

ebookmain.c

分组定义（在一页内分组显示）

分组定义格式：

```
/**
* @name 分组说明文字
* @{
*/
    … 定义的内容 …
/** @} */
```

例：

```
/**
*@name   文件名常量
*@{
*/
#define FILE_NAME   “B:\\test.txt”
/** @}*/

/**
*@name   系统常量
*@{
*/
#define SYS_COUT   5
/** @}*/
```

结束语

谢谢大家！