**[ctags使用细节](http://www.cnblogs.com/david-wei0810/p/5950196.html)**

*在src code目录中运行下面的命令(我自己使用的命令）：*

*$ctags --langmap=c++:.h --languages=c++,c,perl,verilog -R*

其中，指定ctags用特定语言的分析器来分析某种扩展名的文件或者名字符合特定模式的文件。例如--langmap=c++:.h告知ctags，以.h为扩展名的文件是c++文件。

过几秒钟后执行完成，会产生一个tags文件，你可以用less查看下tags的内容，格式比较简单，可以参考其他资料。vim打开一个源文件，把鼠标移到调用函数名上，按住‘CTRL’+‘]’就可以跳转到函数的定义处了。或者使用命令完成相同的功能：

*:tag 标识符*

当有多处匹配标识符时，vim会默认跳转到第一个匹配处。可以使用下列命令进行操作：

* ta[g][!] tagstring  跳转到tagstring定义处，其中！表示在文件更改没保存时进行强制跳转
* tags                查看跳转记录，并用'>'指出当前位置
* [n]ta[g][!]         跳转到相对当前位置的前第n个tag，n默认为1
* [n]po[p][!]         跳转到相对当前位置的后第n个tag，n默认为1
* ts[elect][!] [tagstring]    显示与tagstring的匹配数
* sts[elect][!] [tagstring]   显示与tagstring的匹配数，选中时以子窗口的形式打开
* [n]tn[ext][!]      跳转到相对后n个匹配处
* [n]tp[revious][!]  跳转到相对前n个匹配处
* [n]tr[ewind][!]    跳转到第n个匹配处
* tl[ast][!]         跳转到一个匹配处

注意ctags默认的将.h文件映射成c++文件，如果不加 *--langmap=c++:.h* 。将无法对头文件进行查找，也就是说无法用*:tag tagstring*命令来对定义在头文件里的宏、函数等进行查找。

**一．ctags是干什么的**

ctags的功能：扫描指定的源文件，找出其中所包含的语法元素，并将找到的相关内容记录下来。

我用的是Exuberant Ctags，在Windows上使用，就一个可执行文件，非常绿色，可在sourceforge下载。

**二．ctags可以识别哪些语言，是如何识别的**

ctags识别很多语言，可以用如下命令来查看：

ctags --list-languages

还可以识别自定义语言，具体没研究过。

    ctags是可以根据文件的扩展名以及文件名的形式来确定该文件中是何种语言，从而使用正确的分析器。可以使用如下命令来查看默认哪些扩展名对应哪些语言：

ctags --list-maps

还可以指定ctags用特定语言的分析器来分析某种扩展名的文件或者名字符合特定模式的文件。例如如下命令告知ctags，以inl为扩展名的文件是c++文件。

ctags --langmap=c++:+.inl –R

    并不十分清楚ctags使用何种技术来解析内容，估计包括正则表达式、词法分析、语法分析等等。但ctags不是编译器也不是预处理器，它的解析能力是有限的。例如它虽然可以识别宏定义，但对于使用了宏的语句的识别还是有缺陷的，在一些稍微正规点的代码（例如ACE的库或VC的头文件等）中的某些常规的宏使用方式会导致ctags无法识别，或者识别错误，从而使得ctags没有记录user想记录的内容，或者记录下的信息不准确。另一方面ctags也有聪明的一面，例如在cpp文件中扫描到static的全局变量时，ctags会记录这个变量，而且还会标明说这个变量是局限于本文件的，同样的定义，如果放在h文件中，ctags则不会标明说这个变量是局限于本文件的，因为ctags认为h文件是头文件的一种，会被其他文件include，所以在其他文件中可能会用到该h文件里定义的这个全局变量。

**三．ctags可以识别和记录哪些语法元素**

可以用如下命令查看ctags可以识别的语法元素：

ctags --list-kinds

或者单独查看可以识别的c++的语法元素

ctags --list-kinds=c++

    ctags识别很多元素，但未必全都记录，例如“函数声明”这一语法元素默认是不记录的，可以控制ctags记录的语法元素的种类。如下命令要求ctags记录c++文件中的函数声明和各种外部和前向声明：

ctags -R --c++-kinds=+px

**四．ctags是怎么记录的**

不管一次扫描多少文件，一条ctags命令把记录的内容都记到一个文件里去，默认是当前目录的tags文件，当然这是可以更改的。

每个语法元素对应文件里的一行，学名叫tag entry。

1）开头是tag的名字，其实也就是语法元素的名字，例如记录的是函数的话则tag名就是函数名，记录的是类的话，tag名就是类名。

2）接下来是一个tab。

3）接下来是语法元素所在的文件名。

4）又是一个tab。

5）一条“命令”。这个要解释一下意义：ctags所记录的内容的一个功能就是要帮助像vi这样的编辑器快速定位到语法元素所在的文件中去。前面已经记录了语法元素所在的文件，这条命令的功能就是一旦在vi中打开语法元素所在的文件，并且执行了该“命令”后，vi的光标就能定位到语法元素在文件中的具体位置。所以该“命令”的内容一般分两种，一种是一个正则表达式的搜索命令，一种是第几行的指向命令。默认让ctags在记录时自行选择命令的种类，选择的依据不详，可以通过命令行参数来强制ctags使用某种命令，这里就不多谈了。

6）对于本tag entry（简称tag）所对应的语法元素的描述，例如语法元素的类型等。具体内容和语法元素的种类密切相关。显示哪些描述，显示的格式等都是可以在命令行指定的。例如如下命令要求描述信息中要包含：a表示如果语法元素的类的成员的话，要标明其access（即是public的还是private的）；i表示如果有继承，标明父类；K表示显示语法元素的类型的全称；S表示如果是函数，标明函数的signature；z表示在显示语法元素的类型是使用kind:type的格式。

ctags -R --fields=+aiKSz

    ctags除了记录上述的各种内容之外，还可以在tags文件中记录本次扫描的各个文件，一个文件名对应一个tag entry。默认是不记录的，要强制记录要是使用如下命令：

ctags –R --extra=+f

    还可以强制要求ctags做这样一件事情——如果某个语法元素是类的一个成员，当然ctags默认会给其记录一个tag entry（说白了就是在tags文件里写一行），可以要求ctags对同一个语法元素再记一行。举一个例子来说明：假设语法元素是一个成员函数，ctags默认记录的tag entry中的tag的名字就是该函数的名字（不包括类名作为前缀），而我们强制要求ctags多记的那个tag entry的tag的名字是包含了类明作为前缀的函数的全路径名。这样做有什么好处见下文分析。强制ctags给类的成员函数多记一行的命令为：

ctags -R --extra=+q

**五．vi大概是怎样使用ctags生成的tags文件的**

估计vi是这样使用tags文件的：我们使用vi来定位某个tag时，vi根据我们输入的tag的名字在tags文件中一行行查找，判断每一行tag entry的tag名字（即每行的开头）是否和用户给出的相同，如果相同就认为找到一条记录，最后vi显示所有找到的记录，或者根据这些记录直接跳转到对应文件的特定位置。

考虑到ctags记录的内容和方式，出现同名的tag entry是很常见的现象，例如函数声明和函数定义的tag名字是一样的，重载函数的tag名字是一样的等等。vi只是使用tag名字来搜索，还没智能到可以根据函数的signature来选择相应的tag entry。vi只能简单的显示tag entry的内容给user，让user自行选择。

ctags在记录成员函数时默认是把函数的名字（仅仅是函数的名字，不带任何类名和namespace作为前缀）作为tag的名字的，这样就导致很多不同类但同名的函数所对应的tag entry的名字都是一样的，这样user在vi中使用函数名来定位时就会出现暴多选择，挑选起来十分麻烦。user可能会想在vi中用函数的全路径名来进行定位，但这样做会失败，因为tags文件中没有对应名字的tag entry。要满足用户的这种心思，就要求ctags在记录时针对类的成员多记录一条tag entry，该tag entry和已有的tag entry的内容都相同，除了tag的名字不同，该tag entry的名字是类的成员的全路径名（包括了命名空间和类名）。这就解释了ctags的--extra=+q这样一条命令行选项（见四）。

**六．我的一条ctags命令**

ctags -R --languages=c++ --langmap=c++:+.inl -h +.inl --c++-kinds=+px --fields=+aiKSz --extra=+q --exclude=lex.yy.cc --exclude=copy\_lex.yy.cc

命令太长了，折成两行了，可以考虑把命令的各个参数写到文件里去了（具体做法就不谈了）。

1.

-R

表示扫描当前目录及所有子目录（递归向下）中的源文件。并不是所有文件ctags都会扫描，如果用户没有特别指明，则ctags根据文件的扩展名来决定是否要扫描该文件——如果ctags可以根据文件的扩展名可以判断出该文件所使用的语言，则ctags会扫描该文件。

2.

--languages=c++

只扫描文件内容判定为c++的文件——即ctags观察文件扩展名，如果扩展名对应c++，则扫描该文件。反之如果某个文件叫aaa.py（python文件），则该文件不会被扫描。

3.

--langmap=c++:+.inl

告知ctags，以inl为扩展名的文件是c++语言写的，在加之上述2中的选项，即要求ctags以c++语法扫描以inl为扩展名的文件。

4.

-h +.inl

告知ctags，把以inl为扩展名的文件看作是头文件的一种（inl文件中放的是inline函数的定义，本来就是为了被include的）。这样ctags在扫描inl文件时，就算里面有static的全局变量，ctags在记录时也不会标明说该变量是局限于本文件的（见第一节描述）。

5.

--c++-kinds=+px

记录类型为函数声明和前向声明的语法元素（见第三节）。

6.

--fields=+aiKSz

控制记录的内容（见第四节）。

7.

--extra=+q

让ctags额外记录一些东西（见第四、五节）。

8.

--exclude=lex.yy.cc --exclude=copy\_lex.yy.cc

告知ctags不要扫描名字是这样的文件。还可以控制ctags不要扫描指定目录，这里就不细说了。