仰望星空

Stay hungry，Stay foolish！

个人资料

wwwang89123

访问： 311094次
积分： 4982
等级： BLOG 5
排名： 第3922名

原创： 150篇
转载： 212篇
译文： 0篇
评论： 75条

Objective--C语言预处理命令之条件编译（#ifdef,#else,#endif,#if等）

标签： c语言 debugging objective-c 宏定义 #enif #define

2013-12-20 10:23 1826人阅读 评论(0) 收藏 举报

分类：

OC（7）

预处理过程扫描源代码，对其进行初步的转换，产生新的源代码提供给编译器。可见预处理过程先于编译器对源代码进行处理。

在C语言中，并没有任何内在的机制来完成如下一些功能：在编译时包含其他源文件、定义宏、根据条件决定编译时是否包含某些代码。要完成这些工作，就需要使用预处理程序。尽管在目前绝大多数编译器都包含了预处理程序，但通常认为它们是独立于编译器的。预处理过程读入源代码，检查包含预处理指令的语句和宏定义，并对源代码进行响应的转换。预处理过程还会删除程序中的注释和多余的空白字符。

预处理指令是以#号开头的代码行。#号必须是该行除了任何空白字符外的第一个字符。#后是指令关键字，在关键字和#号之间允许存在任意个数的空白字符。整行语句构成了一条预处理指令，该指令将在编译器进行编译之前对源代码做某些转换。下面是部分预处理指令：

指令用途

#空指令，无任何效果
#include包含一个源代码文件
#define定义宏
#undef取消已定义的宏
#if如果给定条件为真，则编译下面代码
#ifdef如果宏已经定义，则编译下面代码
#ifndef如果宏没有定义，则编译下面代码
#elif如果前面的#if给定条件不为真，当前条件为真，则编译下面代码
#endif结束一个#if……#else条件编译块
#error停止编译并显示错误信息

一、文件包含

#include预处理指令的作用是在指令处展开被包含的文件。包含可以是多重的，也就是说一个被包含的文件中还可以包含其他文件。标准C编译器至少支持八重嵌套包含。

预处理过程不检查在转换单元中是否已经包含了某个文件并阻止对它的多次包含。这样就可以在多次包含同一个头文件时，通过给定编译时的条件来达到不同的效果。例如：

```
#defineAAA
#include"t.c"
```

#undefAAA
#include"t.c"

为了避免那些只能包含一次的头文件被多次包含，可以在头文件中用编译时条件来进行控制。例如：
/*my.h*/
#ifndefMY_H
#defineMY_H
……
#endif

在程序中包含头文件有两种格式：
#include<my.h>
#include"my.h"
第一种方法是用尖括号把头文件括起来。这种格式告诉预处理程序在编译器自带的或外部库的头文件中搜索被包含的头文件。第二种方法是用双引号把头文件括起来。这种格式告诉预处理程序在当前被编译的应用程序的源代码文件中搜索被包含的头文件，如果找不到，再搜索编译器自带的头文件。

采用两种不同包含格式的理由在于，编译器是安装在公共子目录下的，而被编译的应用程序是在它们自己的私有子目录下的。一个应用程序既包含编译器提供的公共头文件，也包含自定义的私有头文件。采用两种不同的包含格式使得编译器能够在很多头文件中区别出一组公共的头文件。

二、宏
宏定义了一个代表特定内容的标识符。预处理过程会把源代码中出现的宏标识符替换成宏定义时的值。宏最常见的用法是定义代表某个值的全局符号。宏的第二种用法是定义带参数的宏，这样的宏可以象函数一样被调用，但它是在调用语句处展开宏，并用调用时的实际参数来代替定义中的形式参数。

1.#define指令
#define预处理指令是用来定义宏的。该指令最简单的格式是：首先神明一个标识符，然后给出这个标识符代表的代码。在后面的源代码中，就用这些代码来替代该标识符。这种宏把程序中要用到的一些全局值提取出来，赋给一些记忆标识符。
#defineMAX_NUM10
intarray[MAX_NUM];
for(i=0;i<MAX_NUM;i++)/*……*/

在这个例子中，对于阅读该程序的人来说，符号MAX_NUM就有特定的含义，它代表的值给出了数组所能容纳的最大元素数目。程序中可以多次使用这个值。作为一种约定，习惯上总是全部用大写字母来定义宏，这样易于把程序红的宏标识符和一般变量标识符区别开来。如果想要改变数组的大小，只需要更改宏定义并重新编译程序即可。
宏表示的值可以是一个常量表达式，其中允许包括前面已经定义的宏标识符。例如：
#defineONE1
#defineTWO2
#defineTHREE(ONE+TWO)
注意上面的宏定义使用了括号。尽管它们并不是必须的。但出于谨慎考虑，还是应该加上括号的。例如：
six=THREE*TWO;
预处理过程把上面的一行代码转换成：
six=(ONE+TWO)*TWO;
如果没有那个括号，就转换成six=ONE+TWO*TWO;了。
宏还可以代表一个字符串常量，例如：
#defineVERSION"Version1.0Copyright(c)2003"

2.带参数的#define指令
带参数的宏和函数调用看起来有些相似。看一个例子：
#defineCube(x)(x)*(x)*(x)
可以时任何数字表达式甚至函数调用来代替参数x。这里再次提醒大家注意括号的使用。宏展开后完全包含在一

对括号中，而且参数也包含在括号中，这样就保证了宏和参数的完整性。看一个用法：
intnum=8+2;
volume=Cube(num);
展开后为(8+2)*(8+2)*(8+2);
如果没有那些括号就变为8+2*8+2*8+2了。
下面的用法是不安全的：
volume=Cube(num++);
如果Cube是一个函数，上面的写法是可以理解的。但是，因为Cube是一个宏，所以会产生副作用。这里的擦书不是简单的表达式，它们将产生意想不到的结果。它们展开后是这样的：
volume=(num++)*(num++)*(num++);
很显然，结果是10*11*12,而不是10*10*10;
那么怎样安全的使用Cube宏呢？必须把可能产生副作用的操作移到宏调用的外面进行：
intnum=8+2;
volume=Cube(num);
num++;

3.#运算符
出现在宏定义中的#运算符把跟在其后的参数转换成一个字符串。有时把这种用法的#称为字符串化运算符。例如：

```
#definePASTE(n)"adhfkj"#n

main()
{
printf("%s ",PASTE(15));
}
```

宏定义中的#运算符告诉预处理程序，把源代码中任何传递给该宏的参数转换成一个字符串。所以输出应该是adhfkj15。

4.##运算符
##运算符用于把参数连接到一起。预处理程序把出现在##两侧的参数合并成一个符号。看下面的例子：

```
#defineNUM(a,b,c)a##b##c
#defineSTR(a,b,c)a##b##c

main()
{
printf("%d ",NUM(1,2,3));
printf("%s ",STR("aa","bb","cc"));
}
```

最后程序的输出为:
123
aabbcc
千万别担心，除非需要或者宏的用法恰好和手头的工作相关，否则很少有程序员会知道##运算符。绝大多数程序员从来没用过它。

三、条件编译指令

条件编译指令将决定那些代码被编译，而哪些是不被编译的。可以根据表达式的值或者某个特定的宏是否被定义来确定编译条件。

1.#if指令
#if指令检测跟在制造另关键字后的常量表达式。如果表达式为真，则编译后面的代码，知道出现#else、#elif或#endif为止；否则就不编译。

2.#endif指令
#endif用于终止#if预处理指令。

```
#defineDEBUG0
main()
{
#ifDEBUG
printf("Debugging ");
#endif
printf("Running ");
}
```

由于程序定义DEBUG宏代表0，所以#if条件为假，不编译后面的代码直到#endif，所以程序直接输出Running。如果去掉#define语句，效果是一样的。

3.#ifdef和#ifndef

```
#defineDEBUG

main()
{
#ifdefDEBUG
printf("yes ");
#endif
#ifndefDEBUG
printf("no ");
#endif
}
```

#ifdefined等价于#ifdef;#if!defined等价于#ifndef

4.#else指令
#else指令用于某个#if指令之后，当前面的#if指令的条件不为真时，就编译#else后面的代码。#endif指令将中指上面的条件块。

```
#defineDEBUG

main()
{
#ifdefDEBUG
printf("Debugging ");
#else
printf("Notdebugging ");
#endif
printf("Running ");
}
```

5.#elif指令
#elif预处理指令综合了#else和#if指令的作用。

```
#defineTWO

main()
{
#ifdefONE
printf("1 ");
```

```
#elifdefinedTWO
printf("2 ");
#else
printf("3 ");
#endif
}
```

程序很好理解，最后输出结果是2。

6.其他一些标准指令

#error指令将使编译器显示一条错误信息，然后停止编译。
#line指令可以改变编译器用来指出警告和错误信息的文件号和行号。
#pragma指令没有正式的定义。编译器可以自定义其用途。典型的用法是禁止或允许某些烦人的警告信息。

补充：

预处理就是在进行编译的第一遍词法扫描和语法分析之前所作的工作。说白了，就是对源文件进行编译前，先对预处理部分进行处理，然后对处理后的代码进行编译。这样做的好处是，经过处理后的代码，将会变的很精短。关于预处理命令中的文件包含（#include），宏定义（#define），书上已经有了详细的说明，在这里就不详述了。这里主要是对条件编译（#ifdef,#else,#endif,#if等）进行说明。以下分3种情况：

1：情况1：

```
#ifdef _XXXX
...程序段1...
#else
...程序段2...
#endif
```

这表明如果标识符_XXXX已被#define命令定义过则对程序段1进行编译；否则对程序段2进行编译。

例：

```
#define NUM
.............
.............
.............
#ifdef NUM
printf("之前NUM有过定义啦！:) ");
#else
printf("之前NUM没有过定义！:( ");
#endif
}
```

如果程序开头有#define NUM这行，即NUM有定义，碰到下面#ifdef NUM的时候，当然执行第一个printf。否则第二个printf将被执行。
我认为，用这种，可以很方便的开启/关闭整个程序的某项特定功能。

2:情况2：

```
#ifndef _XXXX
...程序段1...
#else
...程序段2...
#endif
```

这里使用了#ifndef，表示的是if not def。当然是和#ifdef相反的状况（如果没有定义了标识符_XXXX，那么执行程序段1，否则执行程序段2）。例子就不举了。

3：情况3：

```
#if 常量
...程序段1...
```

#else
...程序段2...
#endif
这里表示，如果常量为真（非0，随便什么数字，只要不是0），就执行程序段1，否则执行程序段2。
我认为，这种方法可以将测试代码加进来。当需要开启测试的时候，只要将常量变1就好了。而不要测试的时候，只要将常量变0。

上一篇 xcode+ APP 打包以及提交apple审核详细流程(新版本更新提交审核)
下一篇 Unity3D学习笔记总结

我的同类文章

OC（7）

- NSUserDefaults性能优化... 2015-06-07 阅读 386
- IOS开发之深拷贝与浅拷贝(... 2013-08-24 阅读 434
- delegate和protocol（协议... 2013-07-04 阅读 592
- Objective-C语法之NSDicti... 2013-03-30 阅读 512
- iOS之nil, Nil, NULL，null和... 2014-10-30 阅读 323
- ios小项目——新浪微博客... 2013-07-05 阅读 728
- Objective-C语法之代码块(... 2013-03-30 阅读 523

查看评论

暂无评论