# 预处理指令

## C 语言执行的流程：

1. 组成程序的每个源文件通过编译过程分别转换成目标代码(object code)
2. 各目标文件由连接器 捆绑在一起，形成一个单一而完整的可执行文件

而编译过程又由如下过程组成：

1. 预处理器处理，在这个阶段，预处理器在源代码上执行一些文本操作。例如：用实际值代替由 #define指令定义的符号以及读入由#include指令包含的文件的内容。
2. 源代码经过解析，判断它的语句的意思，这个阶段会产生绝大多数错误和警告信息的地方，随后便生成目标代码。

执行阶段：

1. 程序必须载入到内存中。
2. 程序执行
3. 程序执行的最后一个阶段就是程序终止。

# Define指令的作用

### 无参宏定义

#define 标识符 字符串

定义常量

意义：就是宏替换的作用，让 标识符替换 字符串，这只是一种简单的文本替换，预处理程序对它不作任何检查。如有错误，只能在编译已被宏展开后的源程序时发现。

反例

#define pint（int\*）

pint pa, pb;

  本意是定义pa和pb均为int型指针，但实际上变成int\* pa,pb;。pa是int型指针，而pb是int型变量。本例中可用typedef来代替define，这样pa和pb就都是int型指针了。因为宏定义只是简单的字符串代换，在预处理阶段完成，而typedef是在编译时处理的，它不是作简单的代换，而是对类型说明符重新命名，被命名的标识符具有类型定义说明的功能

无参宏注意事项：

* 宏名一般用大写字母表示，以便于与变量区别。
* 宏定义末尾不必加分号，否则连分号一并替换。
* 宏定义可以嵌套。
* 可用#undef命令终止宏定义的作用域。
* 使用宏可提高程序通用性和易读性，减少不一致性，减少输入错误和便于修改。如数组大小常用宏定义。
* 预处理是在编译之前的处理，而编译工作的任务之一就是语法检查，预处理不做语法检查。
* 宏定义写在函数的花括号外边，作用域为其后的程序，通常在文件的最开头。
* 字符串" "中永远不包含宏，否则该宏名当字符串处理。
* 宏定义不分配内存，变量定义分配内存。

### 带参宏定义

#define宏名（参数） 字符串

例如：

#define INC(x) x+1

y = INC(5);

反例：

#define SQ(r ) r\*r

Y = SQ(a+b); 结果= a+b\*a+b;

使用案例：

我们最常用的一种替代方案区最大最小值：

#define MAX(x,y) (((x) > (y)) ? (x): (y))

## 文件包含

文件包含命令把指定头文件插入该命令行位置取代该命令行，从而把指定的文件和当前的源程序文件连成一个源文件。

在程序设计中，文件包含是很有用的。一个大程序可以分为多个模块，由多个程序员分别编程。有些公用的符号常量或宏定义等可单独组成一个文件，在其它文件的开头用包含命令包含该文件即可使用。这样，可避免在每个文件开头都去书写那些公用量，从而节省时间，并减少出错。

### #include <> 或者 #include “” 区别

< > 和 “ ” 区别在于： 使用《》 表示在包含文件目录中去查找 （包含目录是由用户在设置环境时设置的include 目录“解决方案管理器 -> 属性-> 配置属性 -> VC++ 目录”），而不是在当前源文件目录去寻找； 使用双引号就是先从当前源文件目录中查找。

## 条件编译

第一种方式

#ifdef 标识符 （或者#if defined）

程序段1

#else

程序段2

#endif

或者

#ifdef 标识符 （或 #if defined 标识符）

程序段

#endif

第二种方式

#ifndef

#else

#endif

第三种方式

#if

#else

#endif

在实践中的作用：

1. 屏蔽跨平台差异

在大规模开发过程中，特别是跨平台和系统的软件里，可以在编译时通过条件编译设置编译环境。

#ifdef WINDOWS

#define MYTYPE long

#else

#define MYTYPE float

#endif

1. 包含程序功能模块

#ifdef FLV

Include “fastleavec”

#endif

当不许向别的用户提供该功能，则在编译之前将首部的FLV 加一下横线即可

1. 开关调试信息

#ifdef DEBUG

Prinft(“device\_open (%d)\n”, file);

#endif

这个DEBUG 就是一个开关。可以在整个工程的一个公用的头文件中，当我们处于debug 阶段的时候就将这个开关打开，当我们程序release 版本里面的时候就将这个开关关闭。

1. 避开硬件的限制

有时一些具体的应用环境的硬件不同，但限于条件本地缺乏这种设备，可绕过硬件直接写出预期结果：

#ifndef TEST

I = dial();

#else

I = 0;

#endif

1. 防止头文件重复包含

头文件可以被头文件或者C文件包含。由于头文件包含可以嵌套，C文件就有可能多次包含同一个文件； 或者不同的C文件都包含同一个头文件，编译时就可能出现重复包含的问题。

#ifndef HEAD\_A

#define HEAD\_A

#include <>

#endif