<https://www.cnblogs.com/engraver-lxw/p/7225232.html>

本文主要介绍va\_start和va\_end的使用及原理。

　　在以前的一篇帖子[Format MessageBox 详解](http://www.cnblogs.com/hanyonglu/archive/2011/04/19/2020738.html)中曾使用到va\_start和va\_end这两个宏，但对它们也只是泛泛的了解。

　　介绍这两个宏之前先看一下C中传递函数的参数时的用法和原理：

**1.在C中，当我们无法列出传递函数的所有实参的类型和数目时,可以用省略号指定参数表**

　　void foo(...);  
　　void foo(parm\_list,...);

　　这种方式和我们以前认识的不大一样，但我们要记住这是C中一种传参的形式，在后面我们就会用到它。

**2.函数参数的传递原理**

　　函数参数是以数据结构:栈的形式存取,从右至左入栈。

　　首先是参数的内存存放格式：参数存放在内存的堆栈段中，在执行函数的时候，从最后一个开始入栈。因此栈底高地址，栈顶低地址，举个例子如下：  
　　　　void func(int x, float y, char z);  
　　那么，调用函数的时候，实参 char z 先进栈，然后是 float y，最后是 int x，因此在内存中变量的存放次序是 x->y->z，因此，从理论上说，我们只要探测到任意一个变量的地址，并且知道其他变量的类型，通过指针移位运算，则总可以顺藤摸瓜找到其他的输入变量。

　　下面是 <stdarg.h> 里面重要的几个宏定义如下：  
　　　　typedef char\* va\_list;  
　　　　void va\_start ( va\_list ap, prev\_param ); /\* ANSI version \*/  
　　　　type va\_arg ( va\_list ap, type );   
　　　　void va\_end ( va\_list ap );   
　　va\_list 是一个字符指针，可以理解为指向当前参数的一个指针，取参必须通过这个指针进行。  
　　<Step 1> 在调用参数表之前，定义一个 va\_list 类型的变量，(假设va\_list 类型变量被定义为ap)；  
　　<Step 2> 然后应该对ap 进行初始化，让它指向可变参数表里面的第一个参数，这是通过 va\_start 来实现的，第一个参数是 ap 本身，第二个参数是在变参表前面紧挨着的一个变量,即“...”之前的那个参数；  
　　<Step 3> 然后是获取参数，调用va\_arg，它的第一个参数是ap，第二个参数是要获取的参数的指定类型，然后返回这个指定类型的值，并且把 ap 的位置指向变参表的下一个变量位置；  
　　<Step 4> 获取所有的参数之后，我们有必要将这个 ap 指针关掉，以免发生危险，方法是调用 va\_end，他是输入的参数 ap 置为 NULL，应该养成获取完参数表之后关闭指针的习惯。说白了，就是让我们的程序具有健壮性。通常va\_start和va\_end是成对出现。

　　例如 int max(int n, ...); 其函数内部应该如此实现：  
　　#include <iostream.h>   
　　void fun(int a, ...)   
　　{   
　　　　int \*temp = &a;

　　　　temp++;　　//评论中反馈有问题，待改中

　　　　for (int i = 0; i < a; ++i)   
　　　　{   
　　　　　　cout << \*temp << endl;   
　　　　　　temp++;   
　　　　}   
　　}  
　　int main()   
　　{   
　　　　int a = 1;   
　　　　int b = 2;   
　　　　int c = 3;   
　　　　int d = 4;   
　　　　fun(4, a, b, c, d);   
　　　　system("pause");   
　　　　return 0;   
　　}

　　Output::   
　　1   
　　2   
　　3   
　　4

**3:获取省略号指定的参数**  
　　在函数体中声明一个va\_list，然后用va\_start函数来获取参数列表中的参数，使用完毕后调用va\_end()结束。像这段代码：   
　　void TestFun(char\* pszDest, int DestLen, const char\* pszFormat, ...)   
　　{   
　　　　va\_list args;   
　　　　va\_start(args, pszFormat); //一定要“...”之前的那个参数  
　　　　\_vsnprintf(pszDest, DestLen, pszFormat, args);   
　　　　va\_end(args);   
　　}

**4.演示如何使用参数个数可变的函数，采用ANSI标准形式**   
　　#include 〈stdio.h〉   
　　#include 〈string.h〉   
　　#include 〈stdarg.h〉   
　　/\*函数原型声明，至少需要一个确定的参数，注意括号内的省略号\*/   
　　int demo( char, ... );   
　　void main( void )   
　　{   
   　　　demo("DEMO", "This", "is", "a", "demo!", "");   
　　}   
　　/\*ANSI标准形式的声明方式，括号内的省略号表示可选参数\*/   
　　int demo( char msg, ... )   
　　{   
       　　/\*定义保存函数参数的结构\*/  
   　　　va\_list argp;   
   　　　int argno = 0;   
   　　　char para;   
　　   　/\*argp指向传入的第一个可选参数，msg是最后一个确定的参数\*/   
   　　　va\_start( argp, msg );   
   　　　while (1)   
       　　{   
        　　　　para = va\_arg( argp, char);   
           　　　if ( strcmp( para, "") == 0 )   
               　　　　break;   
           　　　printf("Parameter #%d is: %s\n", argno, para);   
           　　　argno++;   
　　　　}   
　　　　va\_end( argp );   
　　　　/\*将argp置为NULL\*/  
　　　　return 0;   
　　}

 如由不对的地方，非常欢迎给予指导！

——【感谢】资料来自于<http://www.cnblogs.com/hanyonglu/archive/2011/05/07/2039916.html>