


debug kernel

博客园 | 首页 | 发新随笔 | 发新文章 | 联系 | 订阅  | 管理

随笔: 0 文章: 2 评论: 0 引用: 0

__declspec(align(#))

c++中__declspec用法总结(转的)

1. __declspec(align(16)) struct SS{ int a,b; };

它与#pragma pack()是一对兄弟, 前者规定了对齐的最小值, 后者规定了对齐的最大值。同时出现时, 前者优先级高。__declspec(align())的一个特点是, 它仅仅规定了数据对齐的位置, 而没有规定数据实际占用的内存长度, 当指定的数据被放置在确定的位置之后, 其后的数据填充仍然是按照#pragma pack规定的方式填充的, 这时候类/结构的实际大小和内存格局的规则是这样的: 在__declspec(align())之前, 数据按照#pragma pack规定的方式填充, 如前所述。当遇到__declspec(align())的时候, 首先寻找距离当前偏移向后最近的对齐点 (满足对齐长度为max(数据自身长度,指定值)), 然后把被指定的数据类型从这个点开始填充, 其后的数据类型从它的后面开始, 仍然按照#pragma pack填充, 直到遇到下一个__declspec(align())。当所有数据填充完毕, 把结构的整体对齐数值和__declspec(align())规定的值做比较, 取其中较大的作为整个结构的对齐长度。特别的, 当__declspec(align())指定的数值比对应类型长度小的时候, 这个指定不起作用。

2. #pragma section("segname",read)

\ __declspec(allocate("segname")) int i = 0;

\ int main(){ return 1;};

此关键词必须跟随code_seg,const_seg,data_seg,init_seg,section关键字之后使用, 以上例子使用了section关键字。使用此关键字将告知编译器, 其后的变量间被分配在那个数据段。

3. __declspec(deprecated(MY_TEXT)) void func(int) {}

与#pragma deprecated()相同。此声明后, 如果在同一作用域中使用func(int)函数, 将被提醒c4996警告。

4. __declspec(dllimport) declarator

& __declspec(dllexport) declarator

<	2018年9月						>
日	一	二	三	四	五	六	
26	27	28	29	30	31	1	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	1	2	3	4	5	6	

公告

昵称: debug kernel

园龄: 5年2个月

粉丝: 0

关注: 0

+加关注

搜索

<input type="text"/>	<input type="button" value="找找看"/>
<input type="text"/>	<input type="button" value="谷歌搜索"/>

常用链接

- 我的随笔
- 我的评论
- 我的参与

无须多说，此二关键字用于导入导出外接元素。

5. __declspec(jitintrinsic)

用于标记一个函数或元素为64位公共语言运行时。具体用法未见到。

6. __declspec(naked) int func(formal_parameters) {}

此关键字仅用于x86系统，多用于硬件驱动。此关键字可以使编译器在生成代码时不包含任何注释或标记。仅可以对函数的定义使用，不能用于数据声明、定义，或者函数的声明。

7. __declspec(restrict) float * init(int m, int n) {};

& __declspec(noalias) void multiply(float * a, float * b, float * c) {}; // 优化必用!

__declspec(restrict)仅适用于返回指针的函数声明，如 __declspec(restrict) void *malloc(size_t size);restrict declspec 适用于返回非别名指针的函数。此关键字用于 malloc 的 C 运行时库实现，因为它决不会返回已经在当前程序中使用的指针值（除非您执行某个非法操作，如在内存已被释放之后使用它）。restrict declspec 为编译器提供执行编译器优化的更多信息。对于编译器来说，最大的困难之一是确定哪些指针会与其他指针混淆，而使用这些信息对编译器很有帮助。有必要指出，这是对编译器的一个承诺，编译器并不对其进行验证。如果您的程序不恰当地使用 restrict declspec，则该程序的行为会不正确。__declspec(noalias)也是仅适用于函数，它指出该函数是半纯粹的函数。半纯粹的函数是指仅引用或修改局部变量、参数和第一层间接参数。此 declspec 是对编译器的一个承诺，如果该函数引用全局变量或第二层间接指针参数，则编译器会生成将中断应用程序的代码。

8. class X {

\ __declspec(noinline) int mbrfunc() { return 0; /* will not inline*/ };

在类中声明一个函数不需要内联。

9. __declspec(noreturn) extern void fatal () {}

不需要返回值。

10. void __declspec(nothrow) __stdcall f2();

不存在异常抛出。

11. struct __declspec(novtable) X { virtual void mf(); };

\ struct Y : public X {void mf() {printf_s("In Y\n");}};

此关键字标记的类或结构不能直接实例化，否则将引发AV错误(access violation)。此关键字的声明将阻止编译器对构造和析构函数的vfptr的初始化。可优化编译后代码大小。

12. struct S { int i;

\ void putprop(int j) { i = j; }

\ int getprop() { return i; }

\ __declspec(property(get = getprop, put = putprop)) int the_prop;};

此关键字与C#中get & set属性相同，可定义实现针对一个字段的可读或可写。以上例子，可以使用(如果实例化S为ss)如：ss.the_prop = 156; (此时，ss.i == 156)接着如果：cout<< s.the_prop;(此时将调用getprop，使返回156)。

- 最新评论
- 我的标签

Powered by: 博客园

模板提供: 沪江博客

Copyright ©2018 debug kernel

13. __declspec(selectany) (转)

在MFC, ATL的源代码中充斥着__declspec(selectany)的声明。selectany可以让我们在.h文件中初始化一个全局变量而不是只能放在.cpp中。比如有一个类, 其中有一个静态变量, 那么我们可以在.h中通过类似__declspec(selectany) type class::variable = value; 这样的代码来初始化这个全局变量。既是该.h被多次include, 链接器也会为我们剔除多重定义的错误。对于template的编程会有很多便利。

14. __declspec(thread) int in_One_Thread;

声明in_One_Thread为线程局部变量并具有线程存储时限, 以便链接器安排在创建线程时自动分配的存储。

15. struct __declspec(uuid("00000000-0000-0000-c000-000000000046")) IUnknown;

将具有唯一表示符号的已注册内容声明为一个变量, 可使用__uuidof()调用。

好文要顶

关注我

收藏该文



debug kernel

关注 - 0

粉丝 - 0

+加关注

0

0

» 下一篇: 系统会因为某一线程的优先级低, 而永远不调度这个线程吗?

发表于 2013-07-09 19:42 DEBUG KERNEL 阅读(89) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

昵称: Justry2015

评论内容:

