转 C89 和 C99 标准比较

2013年10月16日 16:34:39　阅读数：12534

注1： GCC支持C99, 通过 --std=c99 命令行参数开启，如：
代码:
gcc --std=c99 test.c

注2： FFMPEG使用的是C99。而VC支持的是C89（不支持C99）。因此VC一般情况下是无法编译FFMPEG的源代码的。

1、增加restrict指针
C99中增加了公适用于指针的restrict类型修饰符，它是初始访问指针所指对象的惟一途径，因此只有借助restrict指针表达式才能访问对象。restrict指针指针主要用做函数变元，或者指向由malloc()函数所分配的内存变量。restrict数据类型不改变程序的语义。
如果某个函数定义了两个restrict指针变元，编译程序就假定它们指向两个不同的对象，memcpy()函数就是restrict指针的一个典型应用示例。C89中memcpy()函数原型如下：
代码:
void *memcpy (void *s1, const void *s2, size_t size);　　如果s1和s2所指向的对象重叠，其操作就是未定义的。memcpy()函数只能用于不重叠的对象。C99中memcpy()函数原型如下：
代码:

void *memcpy(void *restrict s1, const void *restrict s2,size_t size);　　通过使用restrict修饰s1和s2 变元，可确保它们在该原型中指向不同的对象。
2、inline（内联）关键字
内联函数除了保持结构化和函数式的定义方式外,还能使程序员写出高效率的代码.函数的每次调用与返回都会消耗相当大的系统资源,尤其是当函数调用发生在重复次数很多的循环语句中时.一般情况下,当发生一次函数调用时,变元需要进栈,各种寄存器内存需要保存.当函数返回时,寄存器的内容需要恢复。如果该函数在代码内进行联机扩展，当代码执行时，这些保存和恢复操作旅游活动会再发生，而且函数调用的执行速度也会大大加快。函数的联机扩展会产生较长的代码，所以只应该内联对应用程序性能有显著影响的函数以及长度较短的函数。

3、新增数据类型
_Bool
值是0或1。C99中增加了用来定义bool、true以及false宏的头文件夹<stdbool.h>，以便程序员能够编写同时兼容于C与C++的应用程序。在编写新的应用程序时，应该使用<stdbool.h>头文件中的bool宏。
_Complex and _Imaginary
C99标准中定义的复数类型如下：float_Complex; float_Imaginary; double_Complex; double_Imaginary; long double_Complex; long double_Imaginary.
<complex.h>头文件中定义了complex和imaginary宏,并将它们扩展为_Complex和_Imaginary,因此在编写新的应用程序时,应该使用<stdbool.h>头文件中的complex和imaginary宏。
long long int
C99标准中引进了long long int（-(2e63 - 1)至2e63 - 1）和unsigned long long int（0 - 2e64 - 1）。long long int能够支持的整数长度为64位。
4、对数组的增强
可变长数组
C99中,程序员声明数组时,数组的维数可以由任一有效的整型表达式确定,包括只在运行时才能确定其值的表达式,这类数组就叫做可变长数组,但是只有局部数组才可以是变长的.
可变长数组的维数在数组生存期内是不变的,也就是说,可变长数组不是动态的.可以变化的只是数组的大小.
可以使用*来定义不确定长的可变长数组。

数组声明中的类型修饰符
在C99中，如果需要使用数组作为函数变元，可以在数组声明的方括号内使用static关键字，这相当于告诉编译程序，变元所指向的数组将至少包含指定的元素个数。也可以在数组声明的方括号内使用restrict,volatile,const关键字，但只用于函数变元。如果使用restrict，指针是初始访问该对象的惟一途径。如果使用const，指针始终指向同一个数组。使用volatile没有任何意义。
5、单行注释
引入了单行注释标记 "//" , 可以象C++一样使用这种注释了。
6、分散代码与声明
7、预处理程序的修改
ａ、变元列表

宏可以带变元，在宏定义中用省略号（...）表示。内部预处理标识符__VA_ARGS__决定变元将在何处得到替换。例：#define MySum(...) sum(__VA_ARGS__) 语句MySum(k,m,n);将被转换成：sum(k, m, n); 变元还可以包含变元。例： #define compare(compf, ...) compf(__VA_ARGS__) 其中的compare(strcmp,"small", "large"); 将替换成：strcmp("small","large");

ｂ、_Pragma运算符

C99引入了在程序中定义编译指令的另外一种方法：_Pragma运算符。格式如下：

_Pragma("directive")

其中directive是要满打满算的编译指令。_Pragma运算符允许编译指令参与宏替换。

ｃ、内部编译指令

STDCFP_CONTRACT ON/OFF/DEFAULT 若为ON，浮点表达式被当做基于硬件方式处理的独立单元。默认值是定义的工具。

STDCFEVN_ACCESS ON/OFF/DEFAULT 告诉编译程序可以访问浮点环境。默认值是定义的工具。

STDC CX_LIMITED_RANGE ON/OFF/DEFAULT 若值为ON，相当于告诉编译程序某程序某些含有复数的公式是可靠的。默认是OFF。

ｄ、新增的内部宏

__STDC_HOSTED__ 若操作系统存在，则为1

__STDC_VERSION__ 199991L或更高。代表C的版本

__STDC_IEC_599__ 若支持IEC 60559浮点运算，则为1

__STDC_IEC_599_COMPLEX__ 若支持IEC 60599复数运算，则为1

__STDC_ISO_10646__ 由编译程序支持，用于说明ISO/IEC 10646标准的年和月格式：yyymmmL

8、for语句内的变量声明

C99中，程序员可以在for语句的初始化部分定义一个或多个变量，这些变量的作用域仅于本for语句所控制的循环体内。比如：

代码:

```
for(int i=0; i<10; i++){
// do someting ...
}
```

9、复合赋值

C99中，复合赋值中，可以指定对象类型的数组、结构或联合表达式。当使用复合赋值时，应在括弧内指定类型，后跟由花括号围起来的初始化列表；若类型为数组，则不能指定数组的大小。建成的对象是未命名的。

例： double *fp = (double[]) {1.1, 2.2, 3.3};

该语句用于建立一个指向double的指针fp，且该指针指向这个3元素数组的第一个元素。 在文件域内建立的复合赋值只在程序的整个生存期内有效。在模块内建立的复合赋值是局部对象，在退出模块后不再存在。

10、柔性数组结构成员

C99中，结构中的最后一个元素允许是未知大小的数组，这就叫做柔性数组成员，但结构中的柔性数组成员前面必须至少一个其他成员。柔性数组成员允许结构中包含一个大小可变的数组。sizeof返回的这种结构大小不包括柔性数组的内存。包含柔性数组成员的结构用malloc()函数进行内存的动态分配，并且分配的内存应该大于结构的大小，以适应柔性数组的预期大小。

11、指定的初始化符

C99中，该特性对经常使用稀疏数组的程序员十分有用。指定的初始化符通常有两种用法：用于数组，以及用于结构和联合。用于数组的格式：[index] = vol; 其中，index表示数组的下标，vol表示本数组元素的初始化值。

例如： int x[10] = {[0] = 10, [5] = 30}; 其中只有x[0]和x[5]得到了初始化．用于结构或联合的格式如下：

member-name(成员名称)

对结构进行指定的初始化时，允许采用简单的方法对结构中的指定成员进行初始化。

例如： struct example{ int k, m, n; } object = {m = 10,n = 200};

其中，没有初始化k。对结构成员进行初始化的顺序没有限制。

12、printf()和scanf()函数系列的增强

C99中printf()和scanf()函数系列引进了处理long long int和unsigned long long int数据类型的特性。long long int 类型的格式修饰符是ll。在printf()和scanf()函数中，ll适用于d, i, o, u 和x格式说明符。另外，C99还引进了hh修饰符。当使用d, i, o, u和x格式说明符时，hh用于指定char型变元。ll和hh修饰符均可以用于n说明符。

格式修饰符a和A用在printf()函数中时，结果将会输出十六进制的浮点数。格式如下：[-]0xh, hhhhp + d 使用A格式修饰符时，x和p必须是大写。A和a格式修饰符也可以用在scanf()函数中，用于读取浮点数。调用printf()函数时，允许在%f说明符前加上l修饰符，即%lf，但不起作用。

13、C99新增的库

C89中标准的头文件

<assert.h> 定义宏assert()
<ctype.h> 字符处理
<errno.h> 错误报告
<float.h> 定义与实现相关的浮点值勤
<limits.h> 定义与实现相关的各种极限值
<locale.h> 支持函数setlocale()
<math.h> 数学函数库使用的各种定义
<setjmp.h> 支持非局部跳转
<signal.h> 定义信号值
<stdarg.h> 支持可变长度的变元列表
<stddef.h> 定义常用常数
<stdio.h> 支持文件输入和输出
<stdlib.h> 其他各种声明
<string.h> 支持串函数
<time.h> 支持系统时间函数
C99新增的头文件和库
<complex.h> 支持复数算法
<fenv.h> 给出对浮点状态标记和浮点环境的其他方面的访问
<inttypes.h> 定义标准的、可移植的整型类型集合。也支持处理最大宽度整数的函数（常见）
<iso646.h> 首先在此1995年第一次修订时引进，用于定义对应各种运算符的宏
<stdbool.h> 支持布尔数据类型类型。定义宏bool，以便兼容于C++
<stdint.h> 定义标准的、可移植的整型类型集合。该文件包含在<inttypes.h>中（常见）
<tgmath.h> 定义一般类型的浮点宏
<wchar.h> 首先在1995年第一次修订时引进，用于支持多字节和宽字节函数
<wctype.h> 首先在1995年第一次修订时引进，用于支持多字节和宽字节分类函数
14、__func__预定义标识符
用于指出__func__所存放的函数名，类似于字符串赋值。

15、其它特性的改动
放宽的转换限制

限制	C89标准	C99标准
数据块的嵌套层数	15	127
条件语句的嵌套层数	8	63
内部标识符中的有效字符个数	31	63
外部标识符中的有效字符个数	6	31
结构或联合中的成员个数	127	1023
函数调用中的参数个数	31	127

不再支持隐含式的int规则
删除了隐含式函数声明
对返回值的约束
C99中,非空类型函数必须使用带返回值的return语句.
扩展的整数类型
扩展类型 含义
int16_t 整数长度为精确16位
int_least16_t 整数长度为至少16位
int_fast32_t 最稳固的整数类型,其长度为至少32位
intmax_t 最大整数类型
uintmax_t 最大无符号整数类型
对整数类型提升规则的改进
C89中,表达式中类型为char,short int或int的值可以提升为int或unsigned int类型.
C99中,每种整数类型都有一个级别.例如:long long int 的级别高于int, int的级别高于char等.在表达式中,其级别低于int或unsigned int的任何整数类型均可被替换成int或unsigned int类型.

原文地址： http://blog.pfan.cn/watersky/14051.html

文章标签： C89 C99 比较 ffmpeg VC
个人分类： FFMPEG 纯编程
所属专栏： FFmpeg