第七章. 错误信息

本章列出了编程中可能遇到的致命错误，语法错误，和警告信息。每节包括一个信息的主要说明，和消除错误或警告条件可采取的措施。

致命错误

致命错误立即终止编译。这些错误通常是命令行指定的无效选项的结果。当编译器不能访问一个特定的源包含文件时也产生致命错误。

致命错误信息采用下面的格式：

C51 FATAL-ERROR –
ACTION: <current action>
LINE: <line in which the error is detected>
ERROR: <corresponding error message>
C51 TERMIANTED.

C51 FATAL-ERROR –
ACTION: <current action>
FILE: <file in which the error is detected>
ERROR: <corresponding error message>
C51 TERMIANTED.

下面说明**Action**和**Error**中可能的内容。

Actions

ALLOCATING MEMORY

编译器不能分配足够的存储区来编译指定的源文件。

CREATING LIST-FILE / OBJECT-FILE / WORKFILE

编译器不能建立列表文件，OBJ文件，或工作文件。这个错误的出现可能是磁盘满或写保护，或文件已存在和只读。

GENERATING INTERMEDIATE CODE

源文件包含的一个函数太大，不能被编译器编译成虚拟代码。尝试把函数分小或重新编译。

OPENING INPUT-FILE

编译器不能发现或打开所选的源或包含文件。

PARSING INVOKE-/#PRAGMA-LINE

当在命令行检测到参数计算，或在一个#**pragma**中检测到参数计算，就产生这样的错误。

PARSING SOURCE-FILE / ANALYZING DECLARATIONS

源文件包含太多的外部参考。减少源文件访问的外部变量和函数的数目。

WRITING TO FILE

当写入列表文件，OBJ文件，或工作文件时遇到的错误。

Errors

‘(‘ AFTER CONTROL EXPECTED

一些控制参数需要用括号包含一个参数。当没有左括号时显示本信息。

‘)’ AFTER PARAMETER EXPECTED

本信息表示包含没有参数的右括号。

BAD DIGIT IN NUMBER

一个控制参数的数字参数包含无效字符。只能是十进制数。

CAN’T CREATE FILE

在**FILE**行定义的文件名不能建立。

CAN’T HAVE GERERAL CONTROL IN INVOCATION LINE

一般控制（例如，**EJECT**）不能包含在命令行。把这些控制用**#pragma**声明放在源文件中。

FILE DOES NOT EXIST

没有发现定义在**FILE**行的文件。

FILE WRITE-ERROR

因为磁盘空间不够，写到列表，预打印，工作，或目标文件时出错。

IDENTIFIER EXPECTED

当**DEFINE**控制没有参数时产生本信息。**DEFINE**要求一个参数作为标识符。这和C语言的规则相同。

MEMORY SPACE EXHAUSTED

编译器不能分配足够的存储区来编译指定的源文件。如果始终出现这个信息，应该把源文件分成两个或多个小文件再重新编译。

MORE THAN 100 ERRORS IN SOURCE-FILE

在编译时检测到的错误超过100个。这使编译器终止。

MORE THAN 256 SEGMENTS/EXTERNALS

在一个源文件中的参考超过256个。单个的源文件不能有超过256个函数或外部参考。这是INTEL目标模块格式（OMF-51）的历史的限制。包含标量和/或**bit**声明的函数在OBJ文件中生成两个，有时候三个段定义。

NON-NULL ARGUMENT EXPECTED

所选的控制参数需要用括号包含一个参数（例如，一个文件名或一个数字）。

OUT OF RANGE NUMBER

一个控制参数的数字参数超出范围。例如，**OPTIMIZE**控制只允许数字0到6。值7就将产生本错误信息。

PARSE STACK OVERFLOW

解析堆栈溢出。如果源程序包含很复杂的表达式或如果块的嵌套深度超过31级，就会出现这个错误。

PREPROCESSOR：LINE TOO LONG（32K）

一个中间扩展长度超过32K字符。

PREPROCESSOR：MACROS TOO NESTED

在宏扩展期间，预处理器所用的堆栈太大。这个信息通常表示一个递归的宏定义，但也可表示一个宏嵌套太多。

RESPECIFIED OR CONFLICTING CONTROL

一个命令行参数指定了两次，或命令行参数冲突。

SOURCE MUST COME FROM A DISK-FILE

源和包含文件必须存在。控制台CON:，：CI:，或类似的设备不能作为输入文件。

UNKNOWN CONTROL

所选的控制参数不认识。

语法和语义错误

语法和语义错误一般出现在源程序中。它们确定实际的编程错误。当遇到这些错误时，编译器尝试绕过错误继续处理源文件。当遇到更多的错误时，编译器输出另外的错误信息。但是，不产生OBJ文件。

语法和语义错误在列表文件中生成一条信息。这些错误信息用下面的格式：

***** ERROR *number* IN LINE *line* OF *file*:*error message***

这里：

number 错误号。

line 对应源文件或包含文件的行号。

file 产生错误的源或包含文件名。

error message 对错误的叙述说明。

下表按错误号列出了语法和语义错误。错误信息列出了主要说明和可能的原因和改正。

号	错误信息和说明
100	**跳过不可打印字符0x??** 在源文件中发现一个非法字符。（注意不检查注释中的字符）
101	**字符串没结束** 一个字符串没有用双引号（”）终止。
102	**字符串太长** 一个字符串不能超过**4096**个字符。用串联符号（‘\’）在逻辑上可延长字符串超过4096个字符。这个模式的行终止符在词汇分析时是连续的。
103	**无效的字符常数** 一个字符常数的格式无效。符号‘**\c**’是无效的，除非**c**是任何可打印的ASCII字符。
125	**声明符太复杂（20）** 一个目标的声明可包含最多20个类型修饰符（‘[’，‘]’，‘*’，‘(’，‘)’）。这个错误经常伴随着错误126。

号	错误信息和说明
126	**类型堆栈下溢** 类型声明堆栈下溢。这个错误通常死错误125的副产品。
127	**无效存储类** 一个目标用一个无效的存储空间标识符声明。如果一个目标在一个函数外用存储类**auto**或**register**声明，就会产生本错误。
129	**在‘标记’前缺少‘;’** 本错误通常表示前一行缺少分号。当出现本错误时，编译器会产生很多错误信息。
130	**值超出范围** 在一个**using**或**interrupt**标识符后的数字参数是无效的。**using**标识符要求一个0到3之间的寄存器组号。**interrupt**标识符要求一个0到31之间的中断矢量号。
131	**函数参数重复** 一个函数有相同的参数名。在函数声明中参数名必须是唯一的。
132	**没在正式的参数列表** 一个函数的参数声明用了一个名称没在参数名列表中。例如： **char function(v0,v1,v2)** **char *v0,*v1,*v5;** **/* ‘v5’没在正式列表中 */** **{** **/* … */** **}**
134	**函数的xdata/idata/pdata/data不允许** 函数通常位于**code**存储区，不能在别的存储区运行。函数默认定义为存储类型**code**。
135	**bit的存储类错** **bit**标量的声明可能包含一个**static**或**extern**存储类。**register**或**alien**类是无效的。
136	**变量用了‘void’** **void**类型只允许作为一个不存在的返回值，或一个函数的空参数列表（**void func(void)**），或和一个指针组合（**void ***）。
138	**Interrupt()不能接受或返回值** 一个中断函数被定义了一个或多个正式的参数，或一个返回值。中断函数不能包含调用参数或返回值。
140	**位在非法的存储空间** **bit**标量的定义可以包含可选的存储类型**data**。如果没有存储类型，则默认为**data**，因为位通常在内部数据存储区。当试图对一个**bit**标量定义别的数据类型时会产生本错误。
141	**临近*标志*语法错误：期待*别的标志*，…** 编译器所见的***标志***是错误的。参考所显示的期待的内容。
142	**无效的基地址** 一个**sfr**或**sbit**声明的基地址是错误的。有效的基地址范围在0x80到0xFF之间。如果用符号**基地址^位号**声明，则基地址必须是8的倍数。

号	错误信息和说明
143	**无效的绝对位地址** **sbit**声明中的绝对位地址必须在0x80到0xFF之间。
144	**基地址^位号：无效的位号** **sbit**声明中定义的位号必须在0到7之间。
145	**未知的sfr**
146	**无效sfr** 一个绝对位（基地址^位号）的声明包含一个无效的基地址标识符。基地址必须是已经声明的**sfr**。任何别的名称是无效的。
147	**目标文件太大** 单个目标文件不能超过65535（64K字节-1）。
149	**struct/union包含函数成员** struct或union不能包含一个函数类型的成员。但是，指向函数的指针是可以的。
150	**struct/union包含一个bit成员** 一个union不能包含**bit**类型成员。这是8051的结构决定的。
151	**struct/union自我关联** 一个结构不能包含自己。
152	**位号超出位域** 位域声明中指定的位号超过给定基类的位号。
153	**命名的位域不能为零** 命名的位域为零。只要未命名的位域允许为零。
154	**位域指针** 指向位域的指针不允许。
155	**位域要求char/int** 位域的基类要求**char**或**int**。**unsigned char**和**unsigned int**类型也行。
156	**alien只允许对函数**
157	**alien函数带可变参数** 存储类**alien**只对外部PL/M-51函数允许。符号（**char *,…**）在alien函数中是非法的。PL/M-51函数通常要求一个固定的参数表。
158	**函数包含未命名的参数** 一个函数的参数列表定义包含一个未命名的抽象类型定义。这个符号只允许在函数原型中。
159	**void后面带类型** 函数的原型声明可包含一个空参数列表（例如，**int func(void)**）。在**void**后不能再有类型定义。
160	**void无效** **void**类型只在和指针组合，或作为一个函数的不存在的返回值中是合法的。
161	**忽视了正式参数** 在一个函数内，一个外部函数的声明用了一个没有类型标识符的参数名列表（例如，**extern yylex(a,b,c);**）。

号	错误信息和说明
180	**不能指向一个‘函数’** 指向一个函数的类型是无效的。尝试用指针指向一个函数。
181	**操作数不兼容** 对给定的操作符，至少一个操作数类型是无效的（例如，**~float_type**）。
183	**左值不能修改** 要修改的目标位于**code**存储区或有**const**属性，因此不能修改。
184	**sizeof：非法操作数** **sizeof**操作符不能确定一个函数或位域的大小。
185	**不同的存储空间** 一个目标声明的存储空间和前一个同样目标声明的存储空间不同。
186	**解除参照无效** 一个内部编译器问题会产生本信息。如果本错误重复出现请和技术支持接洽。
187	**不是一个左值** 所需的参数必须是一个可修改的目标地址。
188	**未知目标大小** 因为没有一个数组的维数，或间接通过一个**void**指针，一个目标的大小不能计算。
189	**‘&’对bit/sfr非法** 取地址符（’**&**’）不允许对**bit**目标或特殊函数寄存器（**sfr**）。
190	**‘&’：不是一个左值** 尝试建立一个指针指向一个未知目标。
193	**非法操作类型**
193	**对ptr非法add/sub**
193	**对bit的非法操作**
193	**错误操作数类型** 当对一个给定的操作符用了非法的操作数类型时产生本错误。例如，无效的表达式如：**bit*bit**，ptr+ptr，或ptr**anything*。这个错误信息包括引起错误的操作符。 下面的操作对bit类型的操作数是可行的： ■ 赋值（=） ■ OR/复合OR（|，|=） ■ AND/复合AND（**&**，**&**=） ■ XOR/复合XOR（^，^=） ■ bit比较（= =，**!**=） ■ 取反（~） bit操作数可和别的数据类型在表达式中混用。在这种情况类型转换自动执行。
194	**‘*’间接指向一个未知大小的目标** 间接操作符*不能和void指针合用，因为指针所指的目标的大小是未知的。
195	**‘*’间接非法** *操作符不能用到非指针参数。

号	错误信息和说明
196	**存储空间可能无效** 转换一个常数到一个指针常数产生一个无效的存储空间。例如 char *p=0x91234。
198	**sizeof返回零** sizeof操作符返回一个零。
199	**‘->’的左边要求struct/union指针** ->操作符的左边参数必须是一个**struct**指针或一个**union**指针。
200	**‘.’左边要求struct/union** .操作符的左边参数要求必须是**struct**或**union**类型。
201	**未定义的struct/union** 给定的**struct**或**union**名是未知的。
202	**未定义的标识符** 给定的标识符是未定义的。
203	**错误的存储类（参考名）** 本错误表示编译器的一个问题。如果重复出现请接洽技术支持。
204	**未定义的成员** 给定的一个**struct**或**union**成员名是未定义的。
205	**不能调用一个中断函数** 一个中断函数不能象一个正常函数一样调用。中断的入口和退出代码是特殊的。
207	**参数列表声明为‘void’** 参数列表声明为**void**的函数不能从调用者接收参数。
208	**太多的实参** 函数调用包含太多的实参。
209	**太少的实参** 调用函数包含太少的实参。
210	**太多的嵌套调用** 函数的嵌套调用不能超过10级。
211	**调用不是对一个函数** 一个函数的调用项不是对一个函数或函数指针求值。
212	**间接调用：寄存器的参数不匹配** 通过一个指针的间接函数调用不包含实际的参数。一个例外是当所有的参数可以通过寄存器传递。这是由于Cx51所用的传递参数的方法。被调用的函数名必须是已知的，因为参数写到被调用函数的数据段。但是，对间接调用来说，被调用函数的名称是未知的。
213	**赋值符的左边不是一个左值** 赋值符的左边要求一个可修改目标的地址。
214	**非法指针转换** bit，float或集合类型的目标不能转换为指针。
215	**非法类型转换** struct/union/void不能转换为任何别的类型。

号	错误信息和说明
216	**标号用在非数组中，或维数超出** 一个数组引用包含太大的维数，或目标不是一个数组。
217	**非整数索引** 一个数组的维数表达式必须是**char**，**unsigned char**，**int**，或**unsigned int**类型。别的类型都是非法的。
218	**控制表达式用了void类型** 在一个**while**，**for**，或**do**的限制表达式中不能用类型**void**。
219	**long常数缩减为int** 一个常数表达式的值必须能用一个**int**类型表示。
220	**非法常数表达式** 期望一个常数表达式。目标名，变量或函数不允许出现在常数表达式中。
221	**非常数case/dim表达式** 一个**case**或一个维数（**[ ]**）必须是一个常数表达式。
222	**被零除**
223	**被零取模** 编译器检测到一个被零除或取模。
225	**表达式太复杂，需简化** 一个表达式太复杂，必须分成两个或多个子表达式。
226	**重复的struct/union/enum标记** 一个**struct**，**union**，或**enum**名早已定义。
227	**表示一个union标记** 一个**union**名称早已定义为别的类型。
228	**表示一个struct标记** 一个**struct**名早已定义为别的类型。
229	**表示一个enum标记** 一个**enum**名早已定义为别的类型。
230	**未知的struct/union/enum标记** 指定的**struct**，**union**，或**enum**名未定义。
231	**重复定义** 指定的名称已被定义。
232	**重复标号** 指定的标号已定义。
233	**未定义标号** 表示一个标号未定义。有时候这个信息会在实际的标号的几行后出现。这是所用的未定义标号的搜索方法引起的。
234	**‘{’，堆栈范围溢出（31）** 超过了最多31个嵌套块。超出的嵌套块被忽略。
235	**参数<数字>：不同类型** 函数声明的参数类型和函数原型中的不同。

号	错误信息和说明
236	**参数列表的长度不同** 函数声明中的参数数目和函数原型中的不同。
237	**函数早已定义** 试图声明一个函数体两次。
238	**重复成员**
239	**重复参数** 试图定义一个已存在的**struct**成员或函数参数。
240	**超出128个局部bit** 在一个函数内不能超过128个**bit**标量。
241	**auto段太大** 局部目标所需的空间超过模式的极限。最大的段大小定义如下： **SMALL** 128字节 **COMPACT** 256字节 **LARGE** 65535字节
242	**太多的初始化软件** 初始化软件的数目超过初始化目标的数量。
243	**字符串超出范围** 字符串中的字符数目超出字符串初始化的数目。
244	**不能初始化，错误的类型或类** 试图初始化一个**bit**或**sfr**。
245	**未知的pragma，跳过本行** **#pragma**状态未知，所以整行被忽略。
246	**浮点错误** 当一个浮点参数超出32位的范围就产生本错误。32位IEEE值的范围是：±1.175494E-38到±3.402823E+38。
247	**非地址/常数初始化** 一个有效的初始化表达式必须是一个常数值求值或一个目标名加或减去一个常数。
248	**集合初始化需要大括号** 给定**struct**或**union**初始化缺少大括号（**{}**）。
249	**段<名>：段太大** 编译器检测到一个数据段太大。一个数据段的最大的大小由存储空间决定。
250	**‘\esc’；值超过255** 一个字符串常数中的转义序列超过有效值范围。最大值是255。
252	**非法八进制数** 指定的字符不是一个有效的八进制数。
252	**主要控制放错地方，行被忽略** 主要控制必须被指定在C模块的开头，在任何**#include**命令或声明前。

号	错误信息和说明
253	**内部错误（ASMGEN\CLASS）** 在下列情况下出现本错误： ■ 一个内在函数（例如，**_testbit_**）被错误激活。这种情况是在没有函数原型存在和实参数目或类型错误。对这种原因，必须使用合适的声明文件（**INTRINS.H，STRING.H**）。参考第八章中的**instrinsic**函数。 ■ Cx51确认一个内部一致性问题。请接洽技术支持。
255	**switch表达式有非法类型** 在一个**switch**表达式没有合法的数据类型。
256	**存储模式冲突** 一个包含**alien**属性的函数只能包含模式标识符**small**。函数的参数必须位于内部数据区。这适用于所有的外部**alien**声明和**alien**函数。例如： alien plm_func(char c) large {…} 产生错误256。
257	**alien函数不能重入** 一个包含**alien**属性的函数不能同时包含**reentrant**属性。函数参数不能跳过虚拟堆栈传递。这适用于所有的外部alien声明和**alien**函数。
258	**struct/union成员的存储空间非法** **非法空间的参数被忽略** 一个结构的成员或参数不能包含一个存储类型标识符。但，指针所指的目标可能包含一个存储类型。例如： struct vp{char code c;int xdata i; }; 产生错误258。 struct v1{char c;int xdata *i; }; 是正确的struct声明。
259	**指针：不同的存储空间** 一个空指针被关联到别的不同存储空间的空指针。例如： char xdata *p1; char idata *p2; p1 = p2; /* 不同的存储空间 */
260	**指针断开** 一个空指针被关联到一些常数值，这些值超过了指针存储空间的值范围。例如： char idata *p1 = 0x1234; /* 结果是0x34 */

号	错误信息和说明

261	**reentrant()内有bit**
	一个可重入属性的函数的声明中不能包含bit目标。例如：
	
	int func1(int i1) reentrant {
	bit b1,b2; /* 不允许！ */
	return(i1-1);
	}

262	**‘using/disable’：不能返回bit值**
	用**using**属性声明的函数和禁止中断（**#pragma disable**）的函数不能返回一个**bit**值给调用者。例如：
	
	bit test(void) using 3
	{
	bit b0;
	return(b0);
	}
	
	产生错误262。

263	**保存/恢复：堆栈保存溢出/下溢** **#pragma save**的最大嵌套深度是八级。堆栈的**pragma save**和**restore**工作根据LIFO（后进，先出）规则。
264	**内在的‘<内在的名称>’：声明/激活错误** 本错误表示一个内在的函数错误定义（参数数目或省略号）。如果用标准的**.H**文件就不会产生本错误。确认使用了Cx51所有的**.H**文件。不要尝试对内在的库函数定义自己的原型。
265	**对非重入函数递归调用** 非重入函数不能被递归调用，因为这样会覆盖函数的参数和局部数据。如果需要递归调用，需声明函数为可重入函数。
267	**函数定义需要ANSI类型的原型** 一个函数被带参数调用，但是声明是一个空的参数列表。原型必须有完整的参数类型，这样编译器就可能通过寄存器传递参数，和适合应用的调用机制。
268	**任务定义错误（任务ID/优先级/using）** 任务声明错误。
271	**‘asm/endasm’控制放错地方** asm和endasm声明不能嵌套。endasm要求一个汇编块，前面用asm开头。例如： #pragma asm . . . 汇编指令 . . . #pragma endasm

号	错误信息和说明
272	**‘asm’要求激活SRC控制** 在一个源文件中使用asm和endasm，要求文件用SRC控制编译。那么编译器就会生成汇编源文件，然后可以用A51汇编。
273	**‘asm/endasm’在包含文件中不允许** 在包含文件中不允许asm和endasm。为了调试，在包含文件不能有任何的可执行代码。
274	**非法的绝对标识符** 绝对地址标识符对位目标，函数，和局部函数不允许。地址必须和目标的存储空间一致。例如，下面的声明是无效的，因为间接寻址的范围是0x00到0xFF。 idata int _at_ 0x1000;
278	**常数太大** 当浮点参数超出32位的浮点值范围就产生本错误。32位IEEE值的范围是：±1.175494E-38到±3.402823E+38。
279	多次初始化 试图多次初始化一个目标。
280	没有使用符号/标号/参数 在一个函数中声明了一个符号，标号，或参数，但没有使用。
281	非指针类型转换为指针 引用的程序目标不能转换成一个指针。
282	不是一个SFR引用 本函数调用要求一个SFR作为参数。
283	asmparms：参数不适合寄存器 参数不适合可用的CPU寄存器。

284	**<名称>：在可覆盖空间，函数不再可重入** 一个可重入函数包含对局部变量的明确的存储类型标识符。函数不再完全可重入。
300	**注释未结束** 一个注释没有一个结束符（*/）。
301	**期望标识符** 一个预处理器命令期望一个标识符。
302	**误用#操作符** 字符操作符‘#’没有带一个标识符。
303	**期望正式参数** 字符操作符‘#’没有带一个标识符表示当前所定义的宏的一个正式参数名。
304	**错误的宏参数列表** 宏参数列表没有一个大括号，逗号分开的标识符列表。
305	**string/char 常数未结束** 一个字符串活字符常数是无效的。典型的，后引号丢失。
306	**宏调用未结束** 预处理器在收集和扩展一个宏调用的实际的参数时遇到输入文件的结尾。

号	错误信息和说明
307	**宏‘名称’：参数计算不匹配** 在一个宏调用中，实际的参数数目和宏定义的参数数目不匹配。本错误表示指定了太少的参数。
308	**无效的整数常数表达式** 一个 if/elif 命令的数学表达式包含一个语法错误。
309	**错误或缺少文件名** 在一个 include 命令中的文件名参数是无效的，或没有。
310	**条件嵌套过多（20）** 源文件包含太多的条件编译嵌套命令。最多允许 20 级嵌套。
311	**elif/else 控制放错地方**
312	**endif 控制放错地方** 命令 elif，else，和 endif 只有在 if，ifdef，或 ifndef 命令中是合法的。
313	**不能清除预定义的宏‘名称’** 试图清除一个预定义宏。用户定义的宏可以用#undef 命令删除。预定义的宏不能清除。
314	**#命令语法错误** 在一个预处理器命令中，字符‘#’必须跟一个新行或一个预处理器命令名（例如，if/define/ifdef，…）。
315	**未知的#命令‘名称’** 预处理器命令是未知的。
316	**条件未结束** 到文件结尾，endif 的数目和 if 或 ifdef 的数目不匹配。
318	**不能打开文件‘文件名’** 指定的文件不能打开。
319	**‘文件’不是一个磁盘文件** 指定的文件不是一个磁盘文件。文件不能编辑。
320	**用户自定义的内容** 本错误号未预处理器的#error 命令保留。#error 命令产生错误号 320，送出用户定义的错误内容，终止编译器生成代码。
321	**缺少<字符>** 在一个 **include** 命令的文件名参数中，缺少结束符。例如： #include<stdio.h
325	**正参‘名称’重复** 一个宏的正参只能定义一次。
326	**宏体不能以‘##’开始或结束** ‘##’不能是一个宏体的开始或结束。
327	**宏‘宏名’：超过 50 个参数** 每个宏的参数数目不能超过 50。

警告

警告产生潜在问题的信息，他们可能在目标程序的运行过程中出现。警告不妨碍源文件的编译。

警告在列表文件中生成信息。警告信息用下面的格式：

***** WARNING *number* IN LINE *line* OF *file*: *warning message***

这里：

number	错误号。
line	在源文件或包含文件中的对应行号。
file	错误产生的源或包含文件名。
warning message	警告的内容。

下表按号列出了警告。警告信息包括一个主要的内容和可能的原因和纠正措施。

号	警告信息和说明
173	**缺少返回表达式** 一个函数返回一个除了 int 类型以外的别的类型的值，必须包含一个返回声明，包括一个表达式。为了兼容旧的程序，对返回一个 int 值的函数不作检查。
182	**指针指向不同的目标** 一个指针关联了一个不同类型的地址。
185	**不同的存储空间** 一个目标声明的存储空间和前面声明的同样目标的存储空间不同。
196	**存储空间可能无效** 把一个无效的常数值分配给一个指针。无效的指针常数是 long 或 unsigned long。编译器对指针采用 24 位（3 字节）。低 16 位代表偏移。高 8 位代表选择的存储空间。
198	**sizeof 返回零** 一个目标的大小计算结果为零。如果目标是外部的，或如果一个数组的维数没有全知道，则值是错误的。

号	警告信息和说明
206	**缺少函数原型** 因为没有原型声明，被调用的函数是未知的。调用一个未知的函数通常是危险的，参数的数目和实际要求不一样。如果是这种情况，函数调用不正确。 没有函数原型，编译器不能检查参数的数目和类型。要避免这种警告，应在程序中包含函数的原型。 函数原型必须在函数被调用前声明。注意函数定义自动生成原型。
209	**实参太少** 在一个函数调用中包含太少的实参。
219	**long 常数被缩减为 int** 一个常数表达式的值必须能被一个 int 类型所表示。
245	**未知的 pragma，本行被忽略** #pragma 声明是未知的，因此整行程序被忽略。
258	**struct/union 成员的存储空间方法** 参数的存储空间被忽略 一个结构的成员或一个参数不能指定存储类型。但是，指针所指的目标可以包含一个存储类型。例如： struct vp{ char code c;int xdata i; }; 产生警告 258。 struct v1{ char c;int xdata *i; }; 对 **struct** 是正确的声明。
259	**指针：不同的存储空间** 两个要比较的指针没有引用相同的存储类型的目标。
260	**指针折断** 把一个指针转换为一个更小偏移区的指针。转换会完成，但大指针的偏移会折断来适应小指针。
261	**bit 在重入函数** 一个 reentrant 函数不能包含 bit，因为 bit 标量不能保存在虚拟堆栈中。
265	**‘名称’：对非重入函数递归调用** 发现对一个非重入函数直接递归。这可能是故意的，但对每个独立的情况进行功能性检查（通过生成的代码）。间接递归由连接/定位器检查。

号	警告信息和说明
271	**‘asm/endams’控制放错地方** asm 和 endasm 不能嵌套。endasm 要求一个以 asm 声明开头的汇编块。例如： #pragma asm . . . 汇编指令 . . . #pragma endasm

275	**表达式可能无效**
	编译器检测到一个表达式不生成代码。例如：
	
	void test(void) {
	int i1,i2,i3;
	i1,i2,i3; /* 死表达式 */
	i1 << i3; /* 结果未使用 */
	}

276	**常数在条件表达式**
	编译器检测到一个条件表达式有一个常数值。在大多数情况下是一个输入错误。例如：
	
	void test(void) {
	int i1,i2,i3;
	if(i1 = 1) i2 = 3; /* 常数被赋值 */
	while(i3 = 2); /* 常数被赋值 */
	}

277	**指针有不同的存储空间** 一个 typedef 声明的存储空间冲突。例如： typedef char xdata XCC; /* 存储空间 xdata */ typedef XCC idata PICC; /* 存储空间冲突 */
280	**符号/标号未使用** 一个符号或标号定义但未使用。
307	**宏‘名称’：参数计算不匹配** 一个宏调用的实参的数目和宏定义的参数数目不匹配。表示用了太多的的参数。过剩的参数被忽略。
317	**宏‘名称’：重新定义无效** 一个预定义的宏不能重新定义或清除。参考 138 页的“预定义宏常数”。
322	**未知的标识符** 在一个#if 命令行的标识符未定义（等效为 FALSE）。
323	**期望新行，发现多余字符** 一个#命令行正确，但包含多余的非注释字符。例如： #include <stdio.h> foo

号	警告信息和说明
324	**期望预处理器记号** 期望一个预处理器记号，但输入的是一个新行。例如：#line，这里缺少#line 命令的参数。