Error Messages

This chapter lists all of the error messages that can be produced by the compiler, the assembler **as**, and **make**.

The messages are in alphabetical order, and each is marked to indicate which program generated it (e.g., **cc0**, **ccp**). Each message from the compiler indicates whether it is a *fatal*, *error*, *warning*, or *strict* condition. The compilation phases are **cpp**, the preprocessor; **cc0**, the parser; **cc1**, the code generator; **cc2**, the optimizer; and **cc3**, the disassembler.

A fatal message usually indicates a condition that caused the compiler to terminate execution. Fatal errors from the later phases of compilation often cannot be fixed, and may indicate problems in the compiler.

An error message points to a condition in the source code that **Let's C** cannot resolve. This almost always occurs when the program does something illegal, e.g., has unbalanced braces.

Warning messages point out code that is compilable, but may produce trouble when the program is executed. A strict message refers to a passage in the code that is unorthodox and may not be portable.

add of two non-constants (**as**, error)
: The present expression adds one or more elements that will be relocated.

address out of range (**as**, error)
: **jmp** addresses must be to a place within 127 bytes.

address wraparound (**ld**, fatal)
: A segment of the program has exceeded the size allowed by the microprocessor's architecture.

; after target or macroname (**make**, error)
: A semicolon appeared after a target name or a macro name.

ambiguous reference to "*string*" (**cc0**, error)
: *string* is defined as a member of more than one **struct** or **union**, is referenced via a pointer to one of those **struct**s or **union**s, and there is more than one offset that could be assigned.

argument list has incorrect syntax (**cc0**, error)
: The argument list of a function declaration contains something other than a comma-separated list of formal parameters.

string argument mismatch (**cpp**, error)
: The argument *string* does not match the type declared in the function's prototype. Either the function prototype or the argument should be changed.

array bound must be a constant (**cc0**, error)
: An array's size can be declared only with a constant; you cannot declare an array's size by using a variable. For example, it is correct to say **foo[5]**, but illegal to say

```
bar = 5;
foo[bar];
```

array bound must be positive (**cc0**, error)
: An array must be declared to have a positive number of elements. The array flagged here was declared to have a negative size, e.g., **foo[-5]**.

array bound too large (**cc0**, error)
: The array is too large to be compiled with 16-bit index arithmetic. You should devise a way to divide the array into compilable portions.

array row has 0 length (**cc0**, error)
: This message can be triggered by either of two problems. The first problem is declaring an array to have a length of zero; e.g., **foo[0]**. The second problem is failing to declare the size of a dimension *other than the first* in a multi-dimensional array. C allows you to declare an indefinite number of array elements of *n* bytes each, but you cannot declare *n* array elements of an indefinite length. For example, it is correct say **foo[][5]** but illegal to say **foo[5][]**.

#assert failure (**cpp**, error)
: The condition being tested in a **#assert** statement has failed.

associative expression too complex (**cc1**, fatal)
: An expression that uses associative binary operators (e.g., '+') has too many operators; for example, **i=i1+i2+i3+ . . . +i30;**. You should simplify the expression.

## at beginning of macro (**cpp**, error)
: Macro replacement lists may contain tokens that are separated by **##**, but **##** cannot appear at the beginning or the end of the list. The tokens on either side of the **##** are pasted together into one token.

## at end of macro (**cpp**, error)
: Macro replacement lists may contain tokens that are separated by **##**, but **##** cannot appear at the beginning or the end of the list. The tokens on either side of the **##** are pasted together into one token.

auto "*string*" is not addressable (**cc0**, error)
: The identifier *string* cannot be addressed on the stack, probably because it is an extraordinarily large automatic array. Large automatic arrays should usually be declared global or static, not automatic.

bad argument storage class (**cc0**, error)
: An argument was assigned a storage class that the compiler does not recognize. The only valid storage class is **register**.

bad base type for field (**cc0**, error)
: The expression uses a bitwise operator (i.e., '<<', '>>', '&', or '|') with an incorrect type of variable. A bit field must be declared within a **char**, **unsigned char**, **int**, or **unsigned int**. No other base type is allowed.

bad external storage class (**cc0**, error)
: An **extern** has been declared with an invalid storage class, e.g., **register** or **auto**.

bad field width (**cc0**, error)
: A field width was declared either to be negative or to be larger than the object that holds it. For example, **char foo:9** or **char foo:-1** will trigger this error.

bad filler field width (**cc0**, error)
: A filler field width was declared either to be negative or to be larger than the object that holds it. For example, **char foo:9** or **char foo:-1** will trigger this error.

bad flexible array declaration (**cc0**, error)
: A flexible array is missing an array boundary; e.g., **foo[5][]**. C permits you to declare an indefinite number of array elements of *n* bytes each, but you cannot declare an array to have *n* elements of an indefinite number of bytes each.

bad line number after # (**as**, error)
The present expression uses an incorrect line number after a '#', e.g,. a negative number.

Bad macro name (**make**, error)
A bad macro name was used; for example, a macro name included a control character.

break not in a loop (**cc0**, error)
A **break** occurs that is not inside a loop or a **switch** statement.

call of non function (**cc0**, error)
What the program attempted to call is not a function. Check to make sure that you have not accidentally declared a function as a variable; e.g., typing **char *foo;** when you meant **char *foo();**.

cannot add pointers (**cc0**, error)
The program attempted to add two pointers. **int**s or **long**s may be added to or subtracted from pointers, and two pointers to the same type may be subtracted, but no other arithmetic operations are legal on pointers.

cannot apply unary '&' to a bit field (**cc0**, error)
The program attempted to use the address of a bit within a byte, which is illegal. Only bytes can be addressed, not the bits within them.

cannot apply unary '&' to a register variable (**cc0**, error)
Because register variables are stored within registers, they do not have addresses, which means that the unary **&** operator cannot be used with them.

cannot apply unary '&' to an alien function (**cc0**, error)
The unary '&' operator cannot be used with any function that has been declared to be of type **alien**. **alien** functions cannot be called by pointers.

cannot cast double to pointer (**cc0**, error)
The program attempted to cast a **double** to a pointer. This is illegal.

cannot cast pointer to double (**cc0**, error)
The program attempted to cast a pointer to a **double**. This is illegal.

cannot cast structure or union (**cc0**, error)
The program attempted to cast a **struct** or a **union**. This is illegal.

cannot cast to structure or union (**cc0**, error)
The program attempted to cast a variable to a **union** or **struct**. This is illegal.

string: cannot create (**as**, error)
The assembler cannot create the output file it was requested to create. This often is due to a problem with the output device; check and make sure that it is not full, and that it is working correctly.

string: cannot create (**cpp**, fatal)
The preprocessor **cpp** cannot create the output file *string* that it was asked to create. This often is due to a problem with the output device; check and make sure that it is not full and that it is working correctly.

cannot declare array of functions (**cc0**, error)
For example, the declaration **extern int (*f)[]();** declares **f** to be an array of pointers to functions that return **int**s. Arrays of functions are illegal.

cannot declare flexible automatic array (**cc0**, error)
The program does not explicitly declare the number of elements in an automatic array.

cannot fold this expression (**as**, error)
The assembler cannot fold an expression, e.g., **(ax+bx)**. This can occur for any of several reasons.

cannot initialize fields (**cc0**, error)
The program attempted to initialize bit fields within a structure. This is not supported.

cannot initialize unions (**cc0**, error)
The program attempted to initialize a **union** within its declaration. **union**s cannot be initialized in this way.

cannot move '.' back (**as**, error)
The assembler cannot move the location counter backward, only forward.

string: cannot open (**cpp**, **cc0**, fatal)
The compiler cannot open the file *string* of source code that it was asked to read. **cpp** may not have been told the correct directory in which this file is to be found; check that the file is located correctly, and that the **-I** options, if any, are correct.

cannot open include file *string* (**cpp**, **cc0**, fatal)
The program asked for file *string*, which was not found in the same directory as the source file, nor in the default **include** directory specified by the environmental variable **INCDIR**, nor in any of the directories named in **-I** options given to the **cc** command.

string: cannot reopen (**cc2**, fatal)
The optimizer in **cc2** cannot reopen a file with which it has worked. Make sure that your mass storage device is working correctly and that it is not full.

case not in a switch (**cc0**, error)
The program uses a **case** label outside of a **switch** statement. See the Lexicon entry for **case**.

character constant overflows long (**cc0**, error)
The character constant is too large to fit into a **long**. It should be redefined.

character constant promoted to long (**cc0**, warning)
A character constant has been promoted to a **long**.

class not allowed in structure body (**cc0**, error)
A storage class such as **register** or **auto** was specified within a structure.

compound statement required (**cc0**, error)
A construction that requires a compound statement does not have one, e.g., a function definition, array initialization, or **switch** statement.

conditional stack overflow (**cpp**, fatal)
A series of **#if** expressions is nested so deeply that it overflowed the allotted stack space. You should simplify this code.

constant expression required (**cc0**, error)
The expression used with a **#if** statement cannot be evaluated to a numeric constant. It probably uses a variable in a statement rather than a constant.

constant "*number*" promoted to long (**cc0**, warning)
The compiler promoted a constant in your program to **long**; although this is not strictly illegal, it may create problems when you attempt to port your code to another system, especially if the constant appears in an argument list.

constant used in truth context (**cc0**, strict)
: A conditional expression for an **if**, **while**, or **for** statement has turned out to be always true or always false. For example, **while(1)** will trigger this message.

construction not in Kernighan and Ritchie (**cc0**, strict)
: This construction is not found in *The C Programming Language*; although it can be compiled by **Let's C**, it may not be portable to another compiler.

continue not in a loop (**cc0**, error)
: The program uses a **continue** statement that is not inside a **for** for **while** loop.

data in bssd (**as**, error)
: The program attempted to initialize something in the **bssd** segment, which can contain only uninitialized data.

'.' declared as label (**as**, error)
: The present expression uses as a label, e.g., ".:". This is illegal.

#define argument mismatch (**cpp**, warning)
: The definition of an argument in a **#define** statement does not match its subsequent use. One or the other should be changed.

declarator syntax (**cc0**, error)
: The program used incorrect syntax in a declaration.

default label not in a switch (**cc0**, error)
: The program used a **default** label outside a **switch** construct. See the Lexicon entry for **default**.

divide by zero (**cc0**, warning)
: The program will divide by zero if this code is executed. Although the program can be parsed, this statement may create trouble if executed.

duplicated case constant (**cc0**, error)
: A **case** value can appear only once in a **switch** statement. See the Lexicon entries for **case** and **switch**.

#elif used without #if or #ifdef (**cpp**, error)
: An **#elif** control line must be preceded by an **#if**, **#ifdef**, or **#ifndef** control line.

#elif used after #else (**cpp**, error)
: An **#elif** control line cannot be preceded by an **#else** control line.

#else used without #if or #ifdef (**cpp**, error)
: An **#else** control line must be preceded by an **#if**, **#ifdef**, or **#ifndef** control line.

empty switch (**cc0**, warning)
: A **switch** statement has no **case** labels and no **default** labels. See the Lexicon entry for **switch**.

#endif used without #if or #ifdef (**cpp**, error)
: An **#endif** control line must be preceded by an **#if**, **#ifdef**, or **#ifndef** control line.

EOF in comment (**cpp**, fatal)
: Your source file appears to end in mid-comment. The file of source code may have been truncated, or you failed to close a comment; make sure that each open-comment symbol '/*' is balanced with a close-comment symbol '*/'. Also, be sure that you did not accidentally embed a **<ctrl-Z>** in the line.

EOF in macro *string* invocation (**cpp**, error)
Your source file appears to end in a macro call. The source file may have been truncated, or you may have accidentally embedded a **<ctrl-Z>** in the line.

EOF in midline (**cpp**, warning)
Check to see that your source file has not been truncated accidentally. Also, make sure that you did not accidentally embed a **<ctrl-Z>** in the line.

EOF in string (**cpp** ,error)
Your file appears to end in the middle of a quoted string literal. Check to see that your source file has not been truncated accidentally. Also, check that you did not accidentally embed a **<ctrl-Z>** in the line.

#error: *string* (**cpp**, fatal)
An **#error** control line has been expanded, printing the remaining tokens on the line and terminating the program.

error creating address (**as**, error)
The object generator could not build an address in MS-DOS object format. Please contact Mark Williams Company.

error in #define syntax (**cpp**, error)
The syntax of a **#define** statement is incorrect. See the Lexicon entry for **#define** for more information.

error in enumeration list syntax (**cc0**, error)
The syntax of an enumeration declaration contains an error.

error in expression syntax (**cc0**, error)
The parser expected to see a valid expression, but did not find one.

error in #include syntax (**cpp**, error)
An **#include** directive must be followed by a string enclosed by either quotation marks (" ") or angle brackets (<>). Anything else is illegal.

expected comma (**as**, error)
The assembler expected to find a comma in the present expression, but did not.

expected constant (**as**, error)
The assembler expected to find a constant in the present expression, but did not.

exponent overflow in floating point constant (**cc0**, warning)
The exponent in a floating point constant has overflowed. The compiler has set the constant to the maximum allowable value, with the expected sign.

exponent underflow in floating point constant (**cc0**, warning)
The exponent in a floating point constant has underflowed. The compiler has set the constant to zero, with the expected sign.

expression too complex (**cc1**, fatal)
The code generator cannot generate code for an expression. You should simplify your code.

external syntax (**cc0**, error)
This could be one of several errors, most often a missing '{'.

field too wide (**cc0**, error)
A field must fit within an **int**, so the declared field width must not be greater than 16.

file ends within a comment (**cc0**, error)
The source file ended in the middle of a comment. If the program uses nested comments, it may have mismatched numbers of begin-comment and end-comment markers. If not, the

program began a comment and did not end it, perhaps inadvertently when dividing by **something*, e.g., **a=b/*cd;**.

function cannot return a function (**cc0**, error)
: The function is declared to return another function, which is illegal. A function, however, can return a *pointer* to a function, e.g., **int (*signal(n, a))()**

function cannot return an array (**cc0**, error)
: A function is declared to return an array, which is illegal. A function, however, can return a pointer to a structure or array.

functions cannot be parameters (**cc0**, error)
: The program uses a function as a parameter, e.g., **int q(); x(q);**. This is illegal.

identifier *string* has too many arguments (**cpp**, error)
: Too many actual parameters have been provided.

identifier "*string*" is being redeclared (**cc0**, error)
: The program declares variable *string* to be of two different types. This often is due to an implicit declaration, which occurs when a function is used before it is explicitly declared. Check for name conflicts.

identifier "*string*" is not a label (**cc0**, error)
: The program attempts to **goto** a nonexistent label.

identifier "*string*" is not a parameter (**cc0**, error)
: The variable "*string*" did not appear in the parameter list.

identifier "*string*" is not defined (**cc0**, error)
: The program uses identifier *string* but does not define it.

identifier "*string*" not bound to register (**cc0**, strict)
: **Let's C** allows two variables to be bound to registers. If more than two variables are declared to be of type **register**, the first two will be bound to registers and all others defined as ordinary **auto**s. If a variable is declared **register** but does not fit in a 16-bit register, it is defined as an **auto**.

identifier "*string*" not usable (**cc0**, error)
: *string* is probably a member of a structure or **union** which appears by itself in an expression.

illegal character constant (**cc0**, error)
: A legal character constant consists of a a backslash '\' followed by **a**, **b**, **f**, **n**, **r**, **t**, **v**, **x**, or up to three octal digits.

illegal character (*number* decimal) (**cc0**, error)
: A control character was embedded within the source code. *number* is the decimal value of the character.

illegal # construct (**cc0**, error)
: The parser recognizes control lines of the form #*line_number* (decimal) or #*file_name*. Anything else is illegal.

illegal control line (**cpp**, error)
: A '#' is followed by a word that the compiler does not recognize.

illegal cpp character (*n* decimal) (**cpp**, error)
: The character noted cannot be processed by **cpp**. It may be a control character or a non-ASCII character.

illegal integer constant suffix (**cc0**, error)
Integer constants may be suffixed with **u**, **U**, **l**, or **L** to indicate **unsigned**, **long**, or **unsigned long**.

illegal label "*string*" (**cc0**, error)
The program uses the keyword *string* as a **goto** label. Remember that each label must end with a colon.

illegal operation on "void" type (**cc0**, error)
The program tried to manipulate a value returned by a function that had been declared to be of type **void**.

illegal structure assignment (**cc0**, error)
The structures have different sizes.

illegal subtraction of pointers (**cc0**, error)
A pointer can be subtracted from another pointer only if both point to objects of the same size.

illegal use of a pointer (**cc0**, error)
A pointer was used illegally, e.g., multiplied, divided, or &-ed. You may get the result you want if you cast the pointer to a **long**.

illegal use of a structure or union (**cc0**, error)
You may take the address of a **struct**, access one of its members, assign it to another structure, pass it as an argument, and return. All else is illegal.

illegal use of defined (**cpp**, error)
The construction **defined(***token***)** or **defined** *token* is legal only in **#if**, **#elif**, or **#assert** expressions.

illegal use of floating point (**cc0**, error)
A **float** was used illegally, e.g., in a bit-field structure.

illegal use of "void" type (**cc0**, error)
The program used **void** improperly. Strictly, there are only **void** functions; **Let's C** also supports the cast to **void** of a function call.

illegal use of void type in cast (**cc0**, error)
The program uses a pointer where it should be using a variable.

improper operand pair (**as**, error)
The expression uses one or more improper operands to an instruction. For example, **mov a, b** will trigger this message; the instruction should be rendered **mov ax, b** or **mov a, ax**.

string in #if (**cpp**, error)
A syntax error occurred in a **#if** declaration. *string* describes the error in detail.

= in or after dependency (**make**, error)
An equal sign '**=**' appeared within or followed the definition of a macro name or target file; for example, **OBJ=atod.obj=factor.obj** will produce this error.

inappropriate signed (**cc0**, error)
The **signed** modifier may only be applied to **char**, **short**, **int**, or **long** types.

include stack overflow (**cpp**, fatal)
A set of **#include** statements is nested so deeply that the allotted stack space cannot hold them. Examines the files for a loop. You should try to fold some of the header files into one, instead of having them call each other.

Incomplete line at end of file (**make**, error)
An incomplete line appeared at the end of the **makefile**.

(in|out)(b|) must be DX or constant (**as**, error)
The present expression must use either DX or a constant.

(in|out)(b|) must use AX or AL (**as**, error)
The present expression must use use AX or AL.

inappropriate "alien" modifier (**cc0**, error)
The **alien** type is used to interface C with non-C functions; your program tried to use **alien** as an internal function rather than as a reference to an external function.

inappropriate "long" (**cc0**, error)
Your program used the type **long** inappropriately, e.g., to describe a **char**.

inappropriate "short" (**cc0**, error)
Your program used the type **short** inappropriately, e.g., to describe a **char**.

inappropriate "unsigned" (**cc0**, error)
Your program used the type **unsigned** inappropriately, e.g., to describe a **double**.

index by non-register (**as**, error)
The present expression attempted to index either by using a variable, or by using a non-existent register.

indirection through non pointer (**cc0**, error)
The program attempted to use a scalar (e.g., a **long** or fBint) as a pointer; you must first cast it to a pointer.

initializer too complex (**cc0**, error)
An initializer was too complex to be calculated at compile time. You should simplify the initializer to correct this problem.

integer pointer comparison (**cc0**, strict)
The program compares an integer or **long** with a pointer without casting one to the type of the other. Although this is legal, the comparison may not work on machines with non-integer pointers, e.g., Z8001 or LARGE-model i8086, or on machines with pointers larger than **int**s, e.g., the 68000.

integer pointer pun (**cc0**, strict)
The program assigns a pointer to an integer, or vice versa, without casting the right-hand side of the assignment to the type of the left-hand side. For example,

```
char *foo;
long bar;
foo = bar;
```

Although this is permitted, it is often an error if the integer has less precision than the pointer does, as in LARGE-model programs. Make sure that you properly declare all functions that returns pointers.

internal compiler error (**cc0**, **cc1**, **cc2**, **cc3**, fatal)
The program produced a state that should not happen during compilation. Forward a copy of the program, preferably on a machine-readable medium, to Mark Williams Company, together with the version number of the compiler, the command line used to compile the program, and the system configuration. For immediate advice during business hours, telephone Mark Williams Company.

internal error, c=*number* in expr. (**as**, error)
The assembler has detected a situation that “should not occur”. Please send a copy of the source code that triggered this error to Mark Williams Company. For immediate help during business hours, contact Mark Williams Company.

invalid floating-point register (**as**, error)
The present instruction addresses a floating-point register that does not exist.

invalid identifier (**as**, error)
The present expression uses an invalid identifier, e.g., **39xy**.

invalid index (**as**, error)
The present expression attempted to index in a context where it is illegal.

invalid index register (**as**, error)
The present expression attempted to index using an incorrect register, e.g., **cx**, **dx**, **sp**, **ip**, ***s**, ***l**, ***h**.

invalid local symbol (**as**, error)
The present expression uses an invalid local symbol, e.g., **21f**.

invalid operand (**as**, error)
The program uses an invalid operand, e.g., **mov ax, 39f**.

invalid operand pair (**as**, error)
The expression uses one or two incorrect operands. For example, **mov a, b** will trigger this message; this expression should be rendered **mov ax, b** or **mov a, ax**.

invalid operand type (**as**, error)
The present instruction uses an invalid operand type, e.g., **or cs, cs**’.

invalid symbol (**as**, error)
The present instruction uses an invalid symbol, i.e., either an undefined symbol or a symbol that is illegal, such as an opcode or an assembler instruction.

“*string*” is a enum tag (**cc0**, error)

“*string*” is a struct tag (**cc0**, error)

“*string*” is a union tag (**cc0**, error)
string has been previously declared as a tag name for a **struct**, **union**, or **enum**, and is now being declared as another tag. Perhaps the structure declarations have been included twice.

“*string*” is not a tag (**cc0**, error)
A **struct** or **union** with tag *string* is referenced before any such **struct** or **union** is declared. Check your declarations against the reference.

“*string*” is not a typedef name (**cc0**, error)
string was found in a declaration in the position in which the base type of the declaration should have appeared. *string* is not one of the predefined types or a **typedef** name. See the Lexicon entry on **typedef** for more information.

“*string*” is not an “enum” tag (**cc0**, error)
An **enum** with tag *string* is referenced before any such **enum** has been declared. See the Lexicon entry for **enum** for more information.

class “*string*” [*number*] is not used (**cc0**, strict)
Your program declares variable *string* or *number* but does not use it.

jmp must be direct address (**as**, error)
The **jmp** instruction must be used with a direct address; the present expression violates this rule.

label "*string*" undefined (**cc0**, error)
The program does not declare the label *string*, but it is referenced in a **goto** statement.

left side of "*string*" not usable (**cc0**, error)
The left side of the expression *string* should be a pointer, but is not.

lvalue required (**cc0**, error)
The left-hand value of a declaration is missing or incorrect. See the Lexicon entries for **lvalue** and **rvalue**.

macro body too long (**cpp**, fatal)
The size of the macro in question exceeds 200 bytes, which is the limit designed into the preprocessor. Try to shorten or split the macro.

Macro definition too long (**make**, error)
Macro definitions are limited to 128 characters.

macro expansion buffer overflow in *string* (**cpp**, fatal)
A macro call has expanded into more characters than **cpp** can handle. Try to shorten the macro, or break it up.

macro *string* redefined (**cpp**, error)
The program redefined the macro *string*.

macro *string* requires arguments (**cpp**, error)
The macro calls for arguments that the program has not supplied.

macros nested *number* deep, loop likely (**cpp**, error)
Macros call each other *number* times; you may have inadvertently created an infinite loop. Try to simplify the program.

member "*string*" is not addressable (**cc0**, error)
The array *string* has exceeded the machine's addressing capability. Structure members are addressed with 16-bit signed offsets on most machines.

member "*string*" is not defined (**cc0**, error)
The program references a structure member that has not been declared.

mismatched conditional (**cc0**, error)
In a '?:' expression, the colon and all three expressions must be present.

misplaced ":" operator (**cc1**, error)
The program used a colon without a preceding question mark. It may be a misplaced label.

missing "(" (**cc0**, error)
The **if**, **while**, **for**, and **switch** keywords must be followed by parenthesized expressions.

missing ')' (**as**, error)
The assembler expected to find a right parenthesis in the present expression, but did not.

missing ")" (**cc0**, error)
A right parenthesis ')' is missing anywhere after a left parenthesis '('.

missing "=" (**cc0**, warning)
An equal sign is missing from the initialization of a variable declaration. Note that this is a warning, not an error: this allows **Let's C** to compile programs with "old style" initializers, such as **int i 1**. Use of this feature is strongly discouraged, and it will disappear when the

draft ANSI standard for the C language is adopted in full.

missing "," (**cc0**, error)
: A comma is missing from an enumeration member list.

missing ":" (**cc0**, error)
: A colon ':' is missing after a **case** label, after a default label, or after the '?' in a '?'-':' construction.

missing ";" (**cc0**, error)
: A semicolon ';' does not appear after an external data definition or declaration, after a **struct** or **union** member declaration, after an automatic data declaration or definition, after a statement, or in a **for(;;)** statement.

missing ']' (**as**, error)
: The assembler expected to find a right bracket in the present expression, but did not.

missing "]" (**cc0**, error)
: A right bracket ']' is missing from an array declaration, or from an array reference; for example, **foo[5**.

missing "{" (**cc0**, error)
: A left brace '{' is missing after a **struct** *tag*, **union** *tag*, or **enum** *tag* in a definition.

missing "}" (**cc0**, error)
: A right brace '}' is missing from a **struct**, **union**, or **enum** definition, from an initialization, or from a compound statement.

missing "while" (**cc0**, error)
: A **while** command does not appear after a **do** in a **do**-**while()** statement.

missing #endif (**cpp**, error)
: An **#if**, **#ifdef**, or **#ifndef** statement was not closed with an **#endif** statement.

missing label name in goto (**cc0**, error)
: A **goto** statement does not have a label.

missing member (**cc0**, error)
: A '.' or '->' is not followed by a member name.

missing output file (**cpp**, fatal)
: The preprocessor **cpp** found a **-o** option that was not followed by a file name for the output file.

missing right brace (**cc0**, error)
: A right brace is missing at end of file. The missing brace probably precedes lines with errors reported earlier.

missing "*string*" (**cc0**, error)
: The parser **cc0** expects to see token *string*, but sees something else.

missing semicolon (**cc0**, error)
: External declarations should continue with ',' or end with ';'.

missing type in structure body (**cc0**, error)
: A structure member declaration has no type.

Multiple actions for *name* (**make**, error)
: A target is defined with more than one single-colon target line.

multiple classes (**cc0**, error)
An element has been asigned to more than one storage class, e.g., **extern register**.

Multiple detailed actions for *name* (**make**, error)
A target is defined with more than one single-colon target line.

multiple #else's (**cpp**, error)
An **#if**, **#ifdef**, or **#ifndef** expression can be followed by no more than one **#else** expression.

multiple types (**cc0**, error)
An element has been assigned more than one data type, e.g., **int float**.

multiply defined symbol (**as**, error)
The current line has re-defined or multiply defined a symbol.

must be CL or 1 (**as**, error)
The present expression must use CL or one, but does not.

must load address into register (**as**, error)
For example, **lea y, x** will trigger this message; this expression should be rendered **lea ax x**.

must load direct address (**as**, error)
For example, **lea ax, (bx, si)** will not work on an i8086.

Must use '::' for *name* (**make**, error)
A double-colon target line was followed by a single-colon target line.

nested comment (**cpp**, warning)
The comment introducer sequence '/*' has been detected within a comment. Comments do not nest.

new line in *string* literal (**cpp**, error)
A newline character appears in the middle of a string. If you wish to embed a newline within a string, use the character constant '\n'. If you wish to continue the string on a new line, insert a backslash '\' before the new line.

Newline after target or macroname (**make**, error)
A newline character appears after a target name or a macro name.

newline in macro argument (**cpp**, warning)
A macro argument contains a newline character. This may create trouble when the program is run.

no 'pop CS' instruction (**as**, error)
The assembler expected to find a 'pop CS' instruction in the present expression, but did not.

no string in .ascii statement (**as**, error)
The present expression uses a **.ascii** instruction, but does not give it a string.

non scalar field (**cc0**, error)
A field must be declared within a **char**, **unsigned char**, **int**, or **unsigned int**. A field with an array base type is not allowed.

non-constant in multiply (**as**, error)
The present expression attempts to multiply one or more elements that will be relocated.

non-constant in segment construction (**as**, error)
The present expression attempts to perform a logical OR on an element that will be relocated.

nonterminated string or character constant (**cc0**, error)
A line that contains single or double quotation marks left off the closing quotation mark. A newline in a string constant may be escaped with '\'.

not a direct address (**as**, error)
For example, **[(bx)+5]** will trigger this message.

not a direct address in this segment (**as**, error)
For example, the expression **[x-y]** is acceptable only if **x** and **y** are in the same segment.

'::' not allowed for *name* (**make**, error)
A double-colon target line was used illegally; for example, after single-colon target line.

number has too many digits (**cc0**, error)
A number is too big to fit into its type.

only one default label allowed (**cc0**, error)
The program uses more than one **default** label in a **switch** expression. See the Lexicon entries for **default** and **switch** for more information.

::: or : in or after dependency list (**make**, error)
A triple colon is meaningless to **make**, and therefore illegal wherever it appears. A single colon may be used only in a target line (which is also called the *dependency list*), and nowhere else.

Out of core (adddep) (**make**, error)
This results from a system problem. Try reducing the size of your **makefile**.

Out of space (**make**, error)
System problem. Try reducing the size of your **makefile**.

Out of space (lookup) (**make**, error)
System problem. Try reducing the size of your **makefile**.

out of space (**cpp**, **cc0**, **cc1**, **cc2**, **cc3**, fatal)
The compiler ran out of space while attempting to compile the program. To remove this error, examine your source and break up any functions that are extraordinarily large.

out of tree space (**cc0**, fatal)
The compiler allows a program to use up to 350 tree nodes; the program exceeded that allowance.

output write error (**cc2**, error)
The optimizer **cc2** cannot create its output file. Check to see if the output device is working correctly, and has enough space to hold the file being created.

parameter *string* is not addressable (**cc0**, error)
The parameter has a stack frame offset greater than 32,767. Perhaps you should pass a pointer instead of a structure.

parameter must follow # (**cpp**, error)
Macro replacement lists may contain **#** followed by a macro parameter name. The macro argument is converted to a string literal.

phase error (**as**, error)
The value of a label changed during the assembly. An expression has a size that differs between the first and second passes.

potentially nonportable structure access (**cc0**, strict)
A program that uses this construction may not be portable to another compiler.

preprocessor assertion failure (**cpp**, warning)
: A **#assert** directive that was tested by the preprocessor **cpp** was found to be false.

string redefined (**cpp**, error)
: **cpp** macros should not be redefined. You should check to see that you are not **#include**ing two different versions of a file somehow, or attempting to use the same macro name for two different purposes.

return type/function type mismatch (**cc0**, error)
: What the function was declared to return and what it actually returns do not match, and cannot be made to match.

return(e) illegal in void function (**cc0**, error)
: A function that was declared to be type **void** has nevertheless attempted to return a value. Either the declaration or the function should be altered.

risky type in truth context (**cc0**, strict)
: The program uses a variable declared to be a pointer, **long**, **unsigned long**, **float**, or **double** as the condition expression in an **if**, **while**, **do**, or '?'-':'. This could be misinterpreted by some C compilers.

segment of improper symbol (**as**, error)
: For example, **es:ax** will trigger this message.

segment override by non-segment register (**as**, error)
: The present expression uses a non-segment register to set
a segment prefix.

size of *string* overflows size_t (**cc0**, strict)
: A string was so large that it overran an internal compiler limit. You should try to break the string in question into several small strings.

size of struct "*string*" is not known (**cc0**, error)
: small strings.

size of union "*string*" is not known (**cc0**, error)
: A pointer to a **struct** or **union** is being incremented, decremented, or subjected to array arithmetic, but the **struct** or **union** has not been defined.

size of *string* too large (**cc0**, error)
: The program declared an array or **struct** that is too big to be addressable, e.g., **long a[20000];** on a machine that has a 64-kilobyte limit on data size and four-byte **long**s.

sizeof truncated to unsigned (**cc0**, warning)
: An object's **sizeof** value has lost precision when truncated to a **size_t** integer.

sizeof(*string*) set to *number* (**cc0**, warning)
: The program attempts to set the value of *string* by applying **sizeof** to a function or an **extern**; the compiler in this instance has set *string* to *number*.

storage class not allowed in cast (**cc0**, error)
: The program **cast**s an item as a **register**, **static**, or other storage class.

string initializer not terminated by NUL (**cc0**, warning)
: An array of **char**s that was initialized by a string is too small in dimension to hold the terminating NUL character. For example, **char foo[3] = "ABC"**.

structure "*string*" does not contain member "*m*" (**cc0**, error)
: The program attempted to address the variable *string.m*, which is not defined as part of the structure *string*.

structure or union used in truth context (**cc0**, error)
: The program uses a structure in an **if**, **while**, or **for**, or ‘?:’ statement.

subtracting non-constant (**as**, error)
: The present expression subtracts one or more elements that will be relocated.

switch of non integer (**cc0**, error)
: The expression in a **switch** statement is not type **int** or **char**. You should cast the **switch** expression to an **int** if the loss of precision is not critical.

switch overflow (**cc1**, fatal)
: The program has more than ten nested **switch**es.

symbol “*string*” truncated to 39 characters (**cc2**, warning)
: A symbol name can have no more than 39 characters.

Syntax error (**make**, error)
: The syntax of a line is faulty.

too many adjectives (**cc0**, error)
: A variable’s type was described with too many of **long**, **short**, or **unsigned**.

too many arguments (**cc0**, fatal)
: No function may have more than 30 arguments.

too many arguments in a macro (**cpp**, fatal)
: The program uses more than the allowed ten arguments with a macro.

too many cases (**cc1**, fatal)
: The program cannot allocate space to build a **switch** statement.

too many directories in include list (**cpp**, fatal)
: The program uses more than the allowed ten **#include** directories.

too many initializers (**cc0**, error)
: The program has more initializers than the space allocated can hold.

Too many macro definitions (**make**, error)
: The number of macros you have created exceeds the capacity of your computer to process them.

too many structure initializers (**cc0**, error)
: The program contains a structure initialization that has more values than members.

trailing “,” in initialization list (**cc0**, warning)
: An initialization statement ends with a comma, which is legal.

type clash (**cc0**, error)
: The parser expected to find matching types but did not. For example, the types of **e1** and **e2** in **(x) ? e1 : e2** must either both be pointers or neither be pointers.

type of function “*string*” adjusted to *string* (**cc0**, warning)
: This warning is given when the type of a numeric constant is widened to **unsigned**, **long**, or **unsigned long** to preserve the constant’s value. The type of the constant may be explicitly specified with the **u** or **L** constant suffixes.

type of parameter “*string*” adjusted to *string* (**cc0**, warning)
: The program uses a parameter that the C language says must be adjusted to a wider type, e.g., **char** to **int** or **float** to **double**.

type required in cast (**cc0**, error)
The type is missing from a cast declaration.

undefined local symbol *string* (**as**, error)
The program uses the symbol *string*, but never defines it.

unexpected end of enumeration list (**cc0**, error)
An end-of-file flag or a right brace occurred in the middle of the list of enumerators.

unexpected end of line (**as**, error)
The present expression ends abruptly, and may have been truncated.

unexpected EOF (**cc0**, **cc1**, **cc2**, **cc3**, fatal)
EOF occurred in the middle of a statement. The temporary file may have been corrupted or truncated accidentally. Check your disk drive to see that it is working correctly. Also, make sure that you did not accidentally embed a **<ctrl-Z>** in the line.

union "*string*" does not contain member *m* (**cc0**, error)
The program attempted to address the variable string *m*, which is not defined as part of the structure *string*.

unknown operator (**as**, error)
The program used an operator that **as** does not recognize.

string: unknown option (**cpp**, fatal)
The preprocessor **cpp** does not recognize the option *string*. Try re-typing the **cc** command line.

= without macro name or in token list (**make**, error)
An equal sign '**=**' can be used only to define a macro, using the following syntax: "MACRO=*definition*". An incomplete macro definition, or the appearance of an equal sign outside the context of a macro definition, will trigger this error message.

: without preceding target (**make**, error)
A colon appeared without a target file name, e.g., :*string*.

write error on output object file (**cc2**, fatal)
cc2 could not write the relocatable object module. Most likely, your mass storage device has run out of room. Check to see that your disk drive or hard disk has enough room to hold the object module, and that it is working correctly.

zero modulus (**cc0**, warning)
The program will perform a modulo operation by zero if the code just parsed is executed. Although the program can be parsed, this statement may create trouble if executed.