

L

L prefix^{C89} A prefix used to introduce a wide character constant or wide string literal.

L suffix For use in a floating-point constant, *see* constant, floating. For use in an integer constant, *see* constant, integer.

l suffix For use in a floating-point constant, *see* constant, floating. For use in an integer constant, *see* constant, integer. For use with math library function names in C89, *see* future library directions.

L10N An abbreviation for localization, a word that has 10 letters between its first and last letter. *See also* I18N.

label A user-defined identifier, followed by a colon, that can appear before a statement. Such a label can be used only in conjunction with a matching `goto` statement. The two other label formats, `case` and `default`, are used only with the `switch` construct and cannot be the object of a `goto`. *See also* labeled statement.

label, case A label involving the `case` keyword that is used only in the context of a `switch` statement. It consists of a translation-time integer constant expression followed by a colon, and it must precede a statement. Duplicate case labels are not permitted within a given switch. A `switch` passes control to a particular `case` label if the controlling expression matches that label's value. *See also* labeled statement.

label, default A label involving the `default` keyword that is used only in the context of a `switch` statement. A switch passes control to the `default` label if the value of the controlling expression does not match any of that switch's `case` labels. *See also* labeled statement.

labeled statement A statement preceded by a label; for example,

```
identifier : statement
case constant-expression : statement
default : statement
```

labs A function that computes the absolute value of its `long int` argument `j`.

```
#include <stdlib.h>
long int labs(long int j);
```

The behavior is undefined if the result cannot be represented. For example, the absolute value of the most negative number cannot be represented in two's complement. On such a system, the absolute value very

likely will be the same as the original value; that is, the absolute value of this most negative value might be negative! *See also* `abs`; `llabs`.

layout The data mapping declared in a structure or union type definition; not to be confused with an actual instance of a structure or union object; for example,

```
struct tag {
    int i;
    double d;
};
```

LC_* macros^{C89} A family of macros, defined in `locale.h`, that expand to distinct integer constant expressions and that are intended for use as the first argument (category) to the `setlocale` function. Standard C has reserved names beginning with `LC_` followed by an uppercase letter for use by implementations so they may add other locale categories. The standard macros are: `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME`. *See also* future library directions.

LC_ALL^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select a whole locale. (The other `LC_*` macros select only a partial locale.) Its use implies all the other `LC_*` categories for that locale.

LC_COLLATE^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select that part of a locale having to do with collation in the functions `strcoll` and `strxfrm`.

LC_CTYPE^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select that part of a locale having to do with character handling and multibyte functions (in `ctype.h` and `stdlib.h`, respectively), and wide characters (in `wchar.h` and `wctype.h`).

LC_MONETARY^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select that part of a locale having to do with the monetary formatting information returned by `localeconv`.

LC_NUMERIC^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select that part of a locale having to do with the selection of a decimal point character in formatted I/O and string conversion functions, and nonmonetary formatting information returned by `localeconv`.

lconv^{C89} A structure type, defined in `locale.h`, that is the key to defining a locale. It contains the following members in any order (other members also may exist):

currency_symbol	int_n_sign_posn ^{C99}	n_cs_precedes
decimal_point	int_p_cs_precedes ^{C99}	n_sep_by_space
frac_digits	int_p_sep_by_space ^{C99}	n_sign_posn
grouping	int_p_sign_posn ^{C99}	positive_sign
int_curr_symbol	mon_decimal_point	p_cs_precedes
int_frac_digits	mon_grouping	p_sep_by_space
int_n_cs_precedes ^{C99}	mon_thousands_sep	p_sign_posn
int_n_sep_by_space ^{C99}	negative_sign	thousands_sep

LC_TIME^{C89} A macro, defined in `locale.h`, that is used as the first argument to `setlocale` to select that part of a locale to do with the behavior of the `strftime` function.

LDBL_DIG^{C89} A macro, defined in `float.h`, that designates the number of decimal digits, such that a `long double` value of that significance can be rounded into a floating-point number and back again without change in those decimal digits.

LDBL_EPSILON^{C89} A macro, defined in `float.h`, that designates the difference between 1.0 and the least value greater than 1.0 that is representable in the `long double` type.

LDBL_MANT_DIG^{C89} A macro, defined in `float.h`, that designates the number of base-`FLT_RADIX` digits in the floating-point significand of a `long double` value.

LDBL_MAX^{C89} A macro, defined in `float.h`, that designates the maximum representable finite `long double` number.

LDBL_MAX_10_EXP^{C89} A macro, defined in `float.h`, that designates the maximum integer such that 10 raised to that power is in a given range of representable finite floating-point numbers.

LDBL_MAX_EXP^{C89} A macro, defined in `float.h`, that designates the maximum integer such that `FLT_RADIX` raised to that power minus 1 is a representable finite floating-point number.

LDBL_MIN^{C89} A macro, defined in `float.h`, that designates the minimum normalized positive `long double` number.

LDBL_MIN_10_EXP^{C89} A macro, defined in `float.h`, that designates the minimum negative integer such that 10 raised to that power is in a given range of normalized floating-point numbers.

LDBL_MIN_EXP^{C89} A macro, defined in `float.h`, that designates the minimum negative integer such that `FLT_RADIX` raised to that power minus 1 is a normalized floating-point number.

ldexp[f|l] A function that multiplies a floating-point number x by 2^{exp} .

```
#include <math.h>
double ldexp(double x, int exp);
float ldexpf(float x, int exp);
long double ldexpl(long double x, int exp);
```

A range error may result. The value returned is $x \times 2^{exp}$.

The `float` and `long double` versions were an invention of C89, where they were optional; however, in C99, they are required.

ldiv^{C89} A function that computes the quotient and remainder when `numer` is divided by `denom`.

```
#include <stdlib.h>
ldiv_t ldiv(long int numer, long int denom);
```

If the division is inexact, the sign of the resulting quotient is that of the algebraic quotient, and the magnitude of the resulting quotient is the largest integer less than the magnitude of the algebraic quotient. If the result cannot be represented, the behavior is undefined. The value returned has the structure type `ldiv_t`, which contains the two `long int` members, `quot` and `rem`, in either order. *See also* `div`; `lldiv`.

ldiv_t^{C89} A type, defined in `stdlib.h`, that is a structure type used as the implementation-defined return type of the `ldiv` function. A possible definition for `ldiv_t` is:

```
typedef struct {
    long int quot;
    long int rem;
} ldiv_t;
```

The ordering of the members does not need to be specified. *See also* `div_t`; `lldiv_t`.

left-shift assignment operator A binary operator, `<<=`, that permits left shift and assignment to be combined such that `exp1 <<= exp2` is equivalent to `exp1 = exp1 << exp2` except that in the former, `exp1` is only evaluated once. Both operands must have integer type, and the left operand must be a modifiable lvalue. The order of evaluation of the operands is unspecified. If the value of the right operand is negative, or equal to or greater than the number of bits in the promoted left operand, the behavior is undefined. The type of the result is the type of `exp1`. This operator associates right to left. *See also* assignment operator, compound.

left-shift operator A binary operator, `<<`, that causes the value of its left operand to be shifted left by the number of bits specified by its right operand. Both operands must have integer type. The order of evaluation of the operands is unspecified. This operator associates left to right. The usual arithmetic conversions are performed on the operands. The result has the type of the left operand after promotion. If the value of the right operand is negative, or equal to or greater than the number of bits in the promoted left operand, the behavior is undefined.

less-than operator A binary operator, `<`, that compares the values of its two operands. Both operands must either have arithmetic type or be pointers to compatible object or incomplete types. The order of evaluation of the operands is unspecified. The result has type `int` and value 0 (if false) or 1 (if true). This operator associates left to right.

less-than-or-equal-to operator A binary operator, `<=`, that compares the values of its operands. Both operands must either have arithmetic type or be pointers to compatible object types or incomplete types. The order of evaluation of the operands is unspecified. The result has type `int` and value 0 (if false) or 1 (if true). This operator associates left to right.

letter One of the 52 uppercase and lowercase alphabetic Latin characters required to be in the execution character set.

lexical element One of a set of fundamental units that, when combined in a proper sequence, make up a valid C program. *See also* token.

lgamma[f|l]^{C99} A function that computes the natural logarithm of the absolute value of gamma of `x`.

```
#include <math.h>
double lgamma(double x);
float lgammaf(float x);
long double lgammal(long double x);
```

If the value of `x` is too large, a range error occurs. A range error may also occur if `x` is nonpositive. The value returned is $\log_e |\Gamma(x)|$.

lifetime A lay term for “storage duration.”

limits, environmental *See* environmental limits.

limits, numerical *See* environmental limits; `float.h`; `limits.h`.

limits, translation *See* environmental limits.

limits.h^{C89,C99} A header that contains a family of macros which describe the integer properties of the target system. (There is also one macro

used for multibyte processing purposes.) While Standard C requires certain minima (or maxima), it is intended that an implementation will document its exact values. The complete set of macros is as follows:

CHAR_BIT	LLONG_MAX ^{C99}	SCHAR_MAX	UINT_MAX
CHAR_MAX	LLONG_MIN ^{C99}	SCHAR_MIN	ULLONG_MAX ^{C99}
CHAR_MIN	LONG_MAX	SHRT_MAX	ULONG_MAX
INT_MAX	LONG_MIN	SHRT_MIN	USHRT_MAX
INT_MIN	MB_LEN_MAX	UCHAR_MAX	

All these macros are guaranteed to expand to a translation-time constant expression suitable for use with `#if`. C99 requires `UCHAR_MAX` to be $2^{\text{CHAR_BIT}} - 1$.

__LINE__ A predefined macro that expands to the current source line number as a decimal constant. (The type of the constant is unspecified but it must have at least the range of `int`.) The following is an example of its use:

```
printf("%lu", (unsigned long)__LINE__);
```

This macro cannot be the subject of `#undef`. *See also* `#line`.

#line A preprocessor directive used to override the current source filename or line number during translation. It is used as follows:

```
#line line-number ["source-file-name"]
```

This directive is not seen often except in code generated under program control by some kind of preprocessor. Standard C allows macro expansion in this directive. *See also* `__FILE__`; `__LINE__`.

line buffered stream A stream in which characters are intended to be sent to or received from the host environment as a block when a new-line is encountered. *See also* `_IOLBF`; `setvbuf`.

line, logical The concatenation of physical source lines that (except for the last) end in a backslash/new-line continuation character.

line, physical A source line terminated by a new-line character. Multiple physical source lines may be concatenated into a logical line.

linkage The form of coupling (if any) between occurrences of the same identifier when used as an object or function name. Standard C defines three forms of linkage: none, internal, and external. Examples of these three forms are an automatic variable, a static function, and external variable, respectively. *See also* `scope`; `storage duration`.

linkage, external A form of linkage in which two or more declarations of a given identifier designate the same object or function. Such identifiers are “exported” from the object module created by the translator and the objects or functions they designate may be accessed by name from any translation unit in which the identifiers are declared. Examples are `non-static` functions and external variables.

linkage, internal A form of linkage in which an identifier that has file scope and designates an object or function, and is `static`. Such identifiers are not “exported” from the object module created by the translator and the objects or functions they designate cannot be accessed by name from any translation unit other than that in which the identifiers are defined.

linkage, no A situation in which an identifier that designates an object and has neither external nor internal linkage. Examples are formal parameters, identifiers other than those designating objects and functions, and `non-extern` block scope identifiers. Such identifiers are private to the blocks in which they are defined.

LL suffix^{C99} *See* constant, integer.

ll suffix^{C99} *See* constant, integer.

llabs^{C99} A function that computes the absolute value of its `long long int` argument `j`.

```
#include <stdlib.h>
long long int llabs(long long int j);
```

The behavior is undefined if the result cannot be represented. For example, the absolute value of the most negative number cannot be represented in two’s complement. On such a system, the absolute value very likely will be the same as the original value; that is, the absolute value of this most negative value might be negative! *See also* `abs`; `labs`.

lldiv^{C99} A function that computes the quotient and remainder when `numer` is divided by `denom`.

```
#include <stdlib.h>
lldiv_t lldiv(long long int numer, long long int denom);
```

If the division is inexact, the sign of the resulting quotient is that of the algebraic quotient, and the magnitude of the resulting quotient is the largest integer less than the magnitude of the algebraic quotient. If the result cannot be represented, the behavior is undefined. The value returned has the structure type `lldiv_t`, which contains the two `long long int` members, `quot` and `rem`, in either order. *See also* `div`; `ldiv`.

`lldiv_t`^{C99} A type, defined in `stdlib.h`, that is a structure type used as the implementation-defined return type of the `lldiv` function. A possible definition for `lldiv_t` is

```
typedef struct {
    long long int quot;
    long long int rem;
} lldiv_t;
```

The ordering of the members does not need to be specified. *See also* `div_t`; `ldiv_t`.

`LLONG_MAX`^{C99} A macro, defined in `limits.h`, that designates the maximum value for an object of type `long long int`. This value must be at least 9,223,372,036,854,775,807 (64 bits). This macro expands to an integer constant expression suitable for use with a `#if` directive.

`LLONG_MIN`^{C99} A macro, defined in `limits.h`, that designates the minimum value for an object of type `long long int`. This value must be at least -9,223,372,036,854,775,807 (64 bits). This macro expands to an integer constant expression suitable for use with a `#if` directive.

`lrint[f|l]`^{C99} A function that rounds its argument to the nearest integer value, according to the current rounding direction.

```
#include <math.h>
long int lrint(double x);
long int lrintf(float x);
long int lrintl(long double x);
```

A range error may occur.

`llround[f|l]`^{C99} A function that returns the rounded integer value of its argument, rounding halfway cases away from zero, regardless of the current rounding direction.

```
#include <math.h>
long long int llround(double x);
long long int llroundf(float x);
long long int llroundl(long double x);
```

A range error may occur.

local A term pertaining to the scope of an identifier in that it is not visible outside its containing and subordinate blocks. *See also* linkage.

locale^{C89} A description of a cultural (or some other) environment. By default, a C program runs in the "C" locale unless the `setlocale` function has been called (or the implementation's normal operating default locale is other than "C"). In the "C" locale, the `ctype.h` routines have their traditional U.S.–English meaning. When a locale other than "C" is selected, the set of characters qualifying for a particular character type test may be extended to include other implementation-defined characters. For example, implementations running in Western Europe likely will include characters such as ä, ß, æ, Å, Ø, or Ł. Whether these test true with `isalpha` is implementation defined, based on the current locale. *See also* locale-specific behavior; `MB_LEN_MAX`.

locale, mixed^{C89} A composite locale comprised of a set of conventions representing two or more distinct nationalities, cultures, or languages. For example, a Swiss locale might include some German aspects as well as some French or Italian aspects. *See also* category; `LC_*` macros.

localeconv^{C89} A function that causes a structure of type `struct lconv` to be initialized with values corresponding to the current locale so that they can be interrogated by the programmer and used to format currency, floating-point data, etc., as needed. The address of this structure is returned.

```
#include <locale.h>
struct lconv *localeconv(void);
```

Undefined behavior results if the structure pointed to by the return value is modified by the caller.

locale.h^{C89} A header that was created as a place to declare routines and types useful in establishing library run-time environments that support cultural and language representations (called locales) in other than the "U.S.–English" mode provided by traditional C. (This traditional locale is known as the "C" locale.)

This header contains definitions or declarations for the following identifiers:

<i>Name</i>	<i>Purpose</i>
LC_ALL	setlocale category
LC_COLLATE	setlocale category
LC_CTYPE	setlocale category
LC_MONETARY	setlocale category
LC_NUMERIC	setlocale category
LC_TIME	setlocale category
struct lconv	Numeric format structure
localeconv	Setup locale information
NULL	Null pointer constant
setlocale	Establish a new locale

The key to defining a locale is the type `struct lconv`. *See also* future library directions.

locale-specific behavior^{C89} Behavior that may be peculiar to a particular locale. Standard functions that have locale-specific behavior are: `atof`, `atoi`, `atol`, `atoll`, `isalnum`, `isalpha`, `isblank`, `isgraph`, `islower`, `isprint`, `ispunct`, `isspace`, `isupper`, `iswalpha`, `iswblank`, `iswctype`, `iswlower`, `iswprint`, `iswpunct`, `iswspace`, `iswupper`, `strcoll`, `strftime`, `strftime`, `strtod`, `strtof`, `strtoimax`, `strtol`, `strtold`, `strtoll`, `strtouimax`, `strtoul`, `strtoull`, `strxfrm`, `tolower`, `toupper`, `towctrans`, `wctod`, `wctof`, `wctol`, `wctold`, `wctoll`, `wctoul`, and `wctoull`. *See also* `wctrans_t`; `wctype_t`.

localization The process of adapting an internationalized program to a particular cultural environment. Known by its abbreviated name L10N. *See also* I18N.

localtime A function that converts the calendar time pointed to by `timer` into a broken-down time, expressed as local time.

```
#include <time.h>
struct tm *localtime(const time_t *timer);
```

The value returned points to the broken-down time object.

log[f|l] A function that computes the natural logarithm of its argument `x`.

```
#include <math.h>
double log(double x);
float logf(float x);
long double logl(long double x);
```

If the argument is negative, a domain error is raised. If the argument is zero and the logarithm of zero cannot be represented (it could be rep-

resented as $-\infty$, perhaps), a range error occurs. Some implementations might generate a domain error instead.

The `float` and `long double` versions were an invention of C89, where they were optional; however, in C99, they are required.

`log10[f|l]` A function that computes the base-ten logarithm of its argument `x`.

```
#include <math.h>
double log10(double x);
float log10f(float x);
long double log10l(long double x);
```

If the argument is negative, a domain error is raised. If the argument is zero and the logarithm of zero cannot be represented (it could be represented as $-\infty$, perhaps), a range error occurs.

The `float` and `long double` versions were an invention of C89, where they were optional; however, in C99, they are required.

`log1p[f|l]`^{C99} A function that returns $\log_e(1 + x)$.

```
#include <math.h>
double log1p(double x);
float log1pf(float x);
long double log1pl(long double x);
```

If the argument is less than -1 a domain error occurs. If the argument equals -1 a range error may occur.

`log2[f|l]`^{C99} A function that returns $\log_2 x$.

```
#include <math.h>
double log2(double x);
float log2f(float x);
long double log2l(long double x);
```

If the argument is less than zero, a domain error occurs. If the argument equals zero, a range error may occur.

logarithmic and exponential functions The `math.h` functions `exp`, `exp2`, `expm1`, `frexp`, `ldexp`, `lgamma`, `log`, `log10`, `log1p`, `log2`, and `modf`, and their `float` and `long double` counterparts, both floating and complex.

`logb[f|l]`^{C99} A function that returns the signed exponent of `x`.

```
#include <math.h>
double logb(double x);
float logbf(float x);
long double logbl(long double x);
```

If the argument is zero, a domain error may occur.

logical AND operator *See* AND operator, logical.

logical negation operator A unary operator, `!`, that logically negates the value of its operand, which must have scalar type. The result has type `int`. If x is zero, $!x$ is 1. If x is nonzero, $!x$ is 0. This operator associates right to left.

logical OR operator *See* OR operator, logical.

logical source line *See* line, logical.

long A permitted abbreviation for `long int`. Also used in the type `long double`.

long double^{C89} Keywords that are used to represent one of the three floating-point types. (The other two are `float` and `double`.) An object of this type must have at least the same range and precision as `double`. *See also* floating type.

long double _Complex^{C99} *See* complex.

long double _Imaginary^{C99} *See* imaginary.

long double suffix^{C99} *See* constant, floating.

long int A standard integer type. Standard C requires it to be at least 32 bits. A plain `long int` is signed. *See also* integer type.

long int suffix *See* constant, integer.

longjmp A function that restores the program's context (or calling environment) that was saved by a previous call to `setjmp` in a user-defined object of type `jmp_buf`.

```
#include <setjmp.h>
void longjmp(jmp_buf env, int val);
```

`longjmp` does not return to its caller. Rather, it returns to the function that originally called `setjmp` to save the corresponding context. In doing so, it passes `val` back to `setjmp` so `setjmp` can return that value to its caller. If the value of `val` is zero, it is made 1 so that a call directly to

`setjmp` (which returns 0) cannot be confused with `longjmp`'s returning through `setjmp` with a value of 0.

The behavior is undefined if `longjmp` is invoked from a nested signal handler. Do not invoke `longjmp` from an exit handler, such as those registered by the `atexit` function.

`long long`^{C99} A permitted abbreviation for `long long int`.

`long long int`^{C99} A standard integer type. Standard C requires it to be at least 64 bits. A plain `long long int` is signed. *See also* integer type.

`LONG_MAX`^{C89} A macro, defined in `limits.h`, that designates the maximum value for an object of type `long int`. It must be at least 2,147,483,647 (32 bits). This macro expands to an integer constant expression suitable for use with a `#if` directive.

`LONG_MIN`^{C89} A macro, defined in `limits.h`, that designates the minimum value for an object of type `long int`. It must be at least $-2,147,483,647$ (32 bits). This macro expands to an integer constant expression suitable for use with an `#if` directive.

`lrint[f|l]`^{C99} A function that rounds its argument to the nearest integer value, according to the current rounding direction.

```
#include <math.h>
long long int llrint(double x);
long long int llrintf(float x);
long long int llrintl(long double x);
```

`lround[f|l]`^{C99} A function that returns the rounded integer value of its argument, rounding halfway cases away from zero, regardless of the current rounding direction.

```
#include <math.h>
long int lround(double x);
long int lroundf(float x);
long int lroundl(long double x);
```

A range error may occur.

`L_tmpnam` A macro, defined in `stdio.h`, that expands to an integer constant expression that is the size of a character array large enough to contain the temporary filename generated by `tmpnam`.

lvalue An expression that designates an object. The name comes from the fact that an lvalue is often found on the left-hand side of assignments. The most common forms of lvalue are the name of a variable and the

expression `*p` where `p` is a pointer to an object. The operators `[]`, unary `*`, and `->` always produce an lvalue, and the dot operator usually produces one. No other operators produce lvalues. *See also* lvalue, modifiable; lvalue, non-modifiable; rvalue.

lvalue, modifiable^{C89} An lvalue through which a value can be stored. Some operators require modifiable lvalues as operands. They are `++`, `--`, and the left operand of all assignment operators. *See also* lvalue, non-modifiable.

lvalue, non-modifiable^{C89} An lvalue through which a value cannot be stored. Examples include the name of an array and any expression designating a `const` object. *See also* lvalue, modifiable.

