ldiv

The `ldiv` function is similar but works with long integers; it returns an `ldiv_t` structure, which also has `quot` and `rem` members. (The `div_t` and `ldiv_t` types are declared in `<stdlib.h>`.)

Q&A

llabs
lldiv

C99

C99 provides two additional functions. The `llabs` function returns the absolute value of a `long long int` value. `lldiv` is similar to `div` and `ldiv`, except that it divides two `long long int` values and returns an `lldiv_t` structure. (The `lldiv_t` type was also added in C99.)

26.3 The `<time.h>` Header: Date and Time

The `<time.h>` header provides functions for determining the time (including the date), performing arithmetic on time values, and formatting times for display. Before we explore these functions, however, we need to discuss how times are stored. `<time.h>` provides three types, each of which represents a different way to store a time:

- `clock_t`: A time value measured in "clock ticks."
- `time_t`: A compact, encoded time and date (a *calendar time*).
- `struct tm`: A time that has been divided into seconds, minutes, hours, and so on. A value of type `struct tm` is often called a *broken-down time*. Table 26.1 shows the members of the `tm` structure. All members are of type `int`.

Table 26.1 Members of the `tm` Structure

Name	*Description*	*Minimum Value*	*Maximum Value*
`tm_sec`	Seconds after the minute	0	61†
`tm_min`	Minutes after the hour	0	59
`tm_hour`	Hours since midnight	0	23
`tm_mday`	Day of the month	1	31
`tm_mon`	Months since January	0	11
`tm_year`	Years since 1900	0	–
`tm_wday`	Days since Sunday	0	6
`tm_yday`	Days since January 1	0	365
`tm_isdst`	Daylight Saving Time flag	††	††

†Allows for two extra "leap seconds." In C99, the maximum value is 60.
††Positive if Daylight Saving Time is in effect, zero if it's not in effect, and negative if this information is unknown.

These types are used for different purposes. A `clock_t` value is good only for representing a time duration; `time_t` and `struct tm` values can store an entire date and time. `time_t` values are tightly encoded, so they occupy little space. `struct tm` values require much more space, but they're often easier to work with. The C standard states that `clock_t` and `time_t` must be "arithmetic types," but leaves it at that. We don't even know if `clock_t` and `time_t` values are stored as integers or floating-point numbers.

We're now ready to look at the functions in `<time.h>`, which fall into two groups: time manipulation functions and time conversion functions.