NAME

Time::Local - efficiently compute time from local and GMT time

SYNOPSIS

```
$time = timelocal($sec,$min,$hour,$mday,$mon,$year);
$time = timegm($sec,$min,$hour,$mday,$mon,$year);
```

DESCRIPTION

This module provides functions that are the inverse of built-in perl functions `localtime()` and `gmtime()`. They accept a date as a six-element array, and return the corresponding `time(2)` value in seconds since the system epoch (Midnight, January 1, 1970 GMT on Unix, for example). This value can be positive or negative, though POSIX only requires support for positive values, so dates before the system's epoch may not work on all operating systems.

It is worth drawing particular attention to the expected ranges for the values provided. The value for the day of the month is the actual day (ie 1..31), while the month is the number of months since January (0..11). This is consistent with the values returned from `localtime()` and `gmtime()`.

FUNCTIONS

timelocal() and timegm()

This module exports two functions by default, `timelocal()` and `timegm()`.

The `timelocal()` and `timegm()` functions perform range checking on the input $sec, $min, $hour, $mday, and $mon values by default.

timelocal_nocheck() and timegm_nocheck()

If you are working with data you know to be valid, you can speed your code up by using the "nocheck" variants, `timelocal_nocheck()` and `timegm_nocheck()`. These variants must be explicitly imported.

```
use Time::Local 'timelocal_nocheck';

# The 365th day of 1999
print scalar localtime timelocal_nocheck 0,0,0,365,0,99;
```

If you supply data which is not valid (month 27, second 1,000) the results will be unpredictable (so don't do that).

Year Value Interpretation

Strictly speaking, the year should be specified in a form consistent with `localtime()`, i.e. the offset from 1900. In order to make the interpretation of the year easier for humans, however, who are more accustomed to seeing years as two-digit or four-digit values, the following conventions are followed:

- Years greater than 999 are interpreted as being the actual year, rather than the offset from 1900. Thus, 1964 would indicate the year Martin Luther King won the Nobel prize, not the year 3864.
- Years in the range 100..999 are interpreted as offset from 1900, so that 112 indicates 2012. This rule also applies to years less than zero (but see note below regarding date range).
- Years in the range 0..99 are interpreted as shorthand for years in the rolling "current century," defined as 50 years on either side of the current year. Thus, today, in 1999, 0 would refer to 2000, and 45 to 2045, but 55 would refer to 1955. Twenty years from now, 55 would instead refer to 2055. This is messy, but matches the way people currently think about two digit dates. Whenever possible, use an absolute four digit year instead.

The scheme above allows interpretation of a wide range of dates, particularly if 4-digit years are used.

Ambiguous Local Times (DST)

Because of DST changes, there are many time zones where the same local time occurs for two different GMT times on the same day. For example, in the "Europe/Paris" time zone, the local time of 2001-10-28 02:30:00 can represent either 2001-10-28 00:30:00 GMT, **or** 2001-10-28 01:30:00 GMT.

When given an ambiguous local time, the timelocal() function should always return the epoch for the *earlier* of the two possible GMT times.

Non-Existent Local Times (DST)

When a DST change causes a locale clock to skip one hour forward, there will be an hour's worth of local times that don't exist. Again, for the "Europe/Paris" time zone, the local clock jumped from 2001-03-25 01:59:59 to 2001-03-25 03:00:00.

If the `timelocal()` function is given a non-existent local time, it will simply return an epoch value for the time one hour later.

IMPLEMENTATION

These routines are quite efficient and yet are always guaranteed to agree with `localtime()` and `gmtime()`. We manage this by caching the start times of any months we've seen before. If we know the start time of the month, we can always calculate any time within the month. The start times are calculated using a mathematical formula. Unlike other algorithms that do multiple calls to `gmtime()`.

The `timelocal()` function is implemented using the same cache. We just assume that we're translating a GMT time, and then fudge it when we're done for the timezone and daylight savings arguments. Note that the timezone is evaluated for each date because countries occasionally change their official timezones. Assuming that `localtime()` corrects for these changes, this routine will also be correct.

BUGS

The whole scheme for interpreting two-digit years can be considered a bug.

SUPPORT

Support for this module is provided via the datetime@perl.org email list. See http://lists.perl.org/ for more details.

Please submit bugs to the CPAN RT system at http://rt.cpan.org/NoAuth/ReportBug.html?Queue=Time-Local or via email at bug-time-local@rt.cpan.org.

COPYRIGHT

AUTHOR

This module is based on a Perl 4 library, timelocal.pl, that was included with Perl 4.036, and was most likely written by Tom Christiansen.

The current version was written by Graham Barr.

It is now being maintained separately from the Perl core by Dave Rolsky, <autarch@urth.org>.