NAME

Date::Manip::Misc − Miscellaneous information about Date::Manip

SHOULD I USE DATE::MANIP

If you look in CPAN, you'll find that there are a number of Date and Time packages. Is Date::Manip the one you should be using? That isn't a trivial question to answer. It depends to a large extent on what you are trying to do.

Date::Manip is certainly one of the most powerful of the Date modules (the other main contender being the DateTime suite of modules). I'm trying to build a library which can do _EVERY_ conceivable date/time manipulation that you'll run into in everyday life dealing with the Gregorian calendar. To the best of my knowledge, it will do everything that any other date module will do which work with the Gregorian calendar, and there are a number of features that Date::Manip has that other modules do not have.

There is a tradeoff in being able to do ''everything''... and that tradeoff is primarily in terms of performance. Date::Manip is written entirely in Perl and is the largest of the date modules. Other modules tend to be faster than Date::Manip, and modules written in C are significantly faster than their Perl counterparts (at least if they're done right). Although I am working on making Date::Manip faster, it will never be as fast as other modules. And before anyone asks, Date::Manip will never be translated to C (at least by me). I write C because I have to. I write Perl because I like to. Date::Manip is something I do because it interests me, not something I'm paid for.

If you are going to be using the module in cases where performance is an important factor, and you're doing a fairly small set of simple date operations over and over again, you should carefully examine the other date modules to see if they will meet your needs.

Date::Manip does NOT provide functionality for working with alternate calendars such as the Chinese or Hebrew calendars, so if you need that functionality, you definitely need to look elsewhere (the DateTime suite probably).

On the other hand, if you want one solution for all your date needs, don't need peak speed, or are trying to do more exotic date operations, Date::Manip is for you. Operations on things like business dates, foreign language dates, holidays and other recurring events, complete timezone handling, etc. are available more-or-less exclusively in Date::Manip. At the very least, if you want to be able to do these operations, it will require using several other modules, each with it's own interface. Also, when you work with Date::Manip, you work with one author and one module. The DateTime suite currently consists of almost 100 modules and 75 authors.

In addition, I am making significant performance improvements in Date::Manip. Although it will never be as fast as some of the other perl modules, I believe that it is already competitive enough for most purposes, and I continue to look for places where I can improve performance, so performance should improve over time.

YEAR 2000 AND YEAR 2007 DST CHANGE

Did Date::Manip have any problems with Y2K compliance? Did it have any problems with the revised daylight saving time changes made in 2007?

Although Date::Manip will parse many date strings (including dates with 2−digit years), internally they are stored as a 4 digit year, and all operations are performed using this internal representation, so Date::Manip had no problems with the Y2K issue. Of course, applications written which stored the year as 2 digits (whether or not it used Date::Manip) may have had problems, but they were not because of this module.

Similarly for the 2007 changes in daylight saving time made in the United States, Date::Manip was not affected. Date::Manip makes use of the current time zone, but it gets that information from the operating system the application is running on. If the operating system knows about the new daylight saving time rules... so does Date::Manip.

WHAT DATES ARE DATE::MANIP USEFUL FOR?

Date::Manip applies to the Gregorian calendar. It does not support alternative calendars (Hebrew, Mayan, etc.) so if you want to use an alternative calendar, you'll need to look elsewhere.

The Gregorian calendar is a relatively recent innovation. Prior to it, the Julian calendar was in use. The

Julian calendar defined leap years as every 4th year. This led to significant calendar drift over time (since a year is NOT 365.25 days long). It was replaced by the Gregorian calendar which improved the definition of leap years, and at that point, the calendar was adjusted appropriately.

Date::Manip extrapolates the Gregorian calendar back to the year 0001 AD and forward to the year 9999 AD, but that does not necessarily mean that the results are useful. As the world adopted the Gregorian calendar, the dates using the Julian calendar had to be changed to fit to account for the drift that had occurred. As such, the dates produced by Date::Manip in an era where the Julian calendar was in use do not accurately reflect the dates actually in use. In historical context, the Julian calendar was in use until 1582 when the Gregorian calendar was adopted by the Catholic church. Protestant countries did not accept it until later; Germany and Netherlands in 1698, British Empire in 1752, Russia in 1918, etc. Date::Manip is therefore not equipped to truly deal with historical dates prior to about 1600, and between 1600 and 1900, the calendar varied from country to country.

A second problem is that the Gregorian calendar is itself imperfect and at some point may need to be corrected (though it's not clear that this will happen... drift may now be accounted for using leap seconds which means that the Gregorian calendar may be useful indefinitely). No attempt is made to correct for the problems in the Gregorian calendar for a couple reasons. First is that my great great great grandchildren will be long dead before this begins to be a problem, so it's not an immediate concern. Secondly, and even more importantly, I don't know what the correction will be (if any) or when it will be implemented, so I can safely ignore it.

There is some limitation on how dates can be expressed such that Date::Manip can handle them correctly. Date::Manip stores the year internally as a 4−digit number. This is obviously not a limit due to the Gregorian calendar, but I needed a way to store the dates internally, and the 4−digit year was chosen. I realize that the 4−digit limitation does create a time when it will break (quite similar to those who chose a 2−digit representation set themselves up for the Y2K problem). Frankly, I'm not too concerned about this since that date is 8000 years in the future! Date::Manip won't exist then. Perl won't exist then. And it's quite possible that the Gregorian calendar won't exist then. That's a much different situation than the Y2K choice in which programmers chose a representation that would break within the lifetime of the programs they were writing.

Given the 4−digit limitation, Date::Manip definitely can't handle BC dates, or dates past Dec 31, 9999. So Date::Manip works (in theory) during the period Jan 1, 0001 to Dec 31, 9999. There are a few caveats:

Gregorian calendar issue

In practical terms, Date::Manip deals with the Gregorian calendar, and is most useful in the period that that calendar has been, or will be, in effect. As explained above, the Gregorian calendar came into universal acceptance in the early 1900's, and it should remain in use for the foreseeable future.

So... in practical terms, Date::Manip is probably useful from around 1900 through several thousand years from now.

First/last week

In one part of the code (calculating week-of-year values), Date::Manip references dates one week after and one week before the date actually being worked on. As such, dates during the first week in the year 0001 fail (because a week before is in the year 1 BC), and those in the last week in the year 9999 fail (because a week later is in 10,000).

No effort will be made to correct this because the added functionality is simply not that important (to me), especially since the Gregorian calendar doesn't really apply in either instance. To be absolutely safe, I will state that Date::Manip works as described in this manual during the period Feb 1, 0001 to Nov 30, 9999, and I will only support dates within that range (i.e. if you submit a bug using a date that is not in that range, I will will consider myself free to ignore it).

Leap seconds

Date::Manip does NOT make use of the leap seconds in calculating time intervals, so the difference between two times may not be strictly accurate due to the addition of a leap second.

Three-digit years

Date::Manip will parse both 2− and 4−digit years, but it will NOT handle 3 digit years. So, if you store the year as an offset from 1900 (which is 3 digits long as of the year 2000), these will NOT be parseable by Date::Manip. Since the perl functions localtime and gmtime DO return the year as an offset from 1900, the output from these will need to be corrected (probably by adding 1900 to the result) before they can be passed to any Date::Manip routine.

FUTURE IDEAS

A number of changes are being considered for future inclusion in Date::Manip. As a rule, the changes listed below are not finalized, and are open to discussion.

Rewrite parsing for better language support

Currently, all of Date::Manip's parsing is based on English language forms of dates, even if the words have been replaced by the equivalent in some other language.

I am considering rewriting the parsing routines in order to allow date forms that might be used in other languages but do not have a common English equivalent, and to account for the fact that some English formats may not have an equivalent in another language.

Adding granularity

The granularity of a time basically refers to how accurate you wish to treat a date. For example, if you want to compare two dates to see if they are identical at a granularity of days, then they only have to occur on the same day. At a granularity of an hour, they have to occur within an hour of each other, etc.

I'm not sure how useful this would be, but it's one of the oldest unimplemented ideas, so I'm not discarding it completely.

ACKNOWLEDGMENTS

There are many people who have contributed to Date::Manip over the years that I'd like to thank. The most important contributions have come in the form of suggestions and bug reports by users. I have tried to include the name of every person who first suggested each improvement or first reported each bug. These are included in the Date::Manip::Changes5 and Date::Manip::Changes6 documents. The list is simply too long to appear here, but I appreciate their help.

A number of people have made suggestions or reported bugs which are not mentioned in these documents. These include suggestions which have not been implemented and people who have made a suggestion or bug report which has already been suggested/reported by someone else. For those who's suggestions have not yet been implemented, they will be added to the appropriate Changes document when (if) their suggestions are implemented. I keep every single suggestion I've ever received and periodically review the unimplemented ones to see if it's something I'm interested in, so even suggestions made years in the past may still appear in future versions of Date::Manip, and the original requester will be attributed at that point (some of the changes made to Date::Manip 6.00 were based on suggestions 10 years old which never fit in with version 5.xx, but which I knew I wanted to implement). For those who have sent in requests/reports that had been previously made by someone else, thank you too. I'd much rather have a suggestion made twice than not at all.

Thanks to Alan Cezar and Greg Schiedler for paying me to implement the Events_List routine. They gave me the idea, and were then willing to pay me for my time to get it implemented quickly.

I'd also like to thank a couple of authors. Date::Manip has gotten some really good press in a couple of books. Since no one's paying me to write Date::Manip, seeing my module get a good review in a book written by someone else really makes my day. My thanks to Nate Padwardhan and Clay Irving (Programming with Perl Modules — part of the O'Reilly Perl Resource Kit); and Tom Christiansen and Nathan Torkington (The Perl Cookbook). Also, thanks to any other authors who've written about Date::Manip who's books I haven't seen.

I'd also like to thank the people who are maintaining the zoneinfo database (and who replied quickly to several inquiries).

I have borrowed from other modules. I originally borrowed the code for determining if a year was a leap

year from code written by David Muir Sharnoff. I borrowed many of the original date printf formats from code written by Terry McGonigal as well as the Solaris date command. More recently, I borrowed the code to do time zone registry lookups on Windows from the DateTime-TimeZone module, though I rewrote it to work better with Date::Manip.

BUGS AND QUESTIONS

Please refer to the Date::Manip::Problems documentation for information on submitting bug reports or questions to the author.

SEE ALSO

Date::Manip − main module documentation

LICENSE

AUTHOR

Sullivan Beck (sbeck@cpan.org)