NAME

Date::Manip::Problems − problems and bugs

KNOWN PROBLEMS

The following are not bugs in Date::Manip, but they may give some people problems.

Unable to determine Time Zone

If you ever get the message that Date::Manip was unable to determine the timezone, you need to provide that information to your script. Please refer to the Date::Manip::TZ documentation for a discussion of this problem.

Calculations appear to be off by an hour

Due to daylight saving time (especially the spring change where the time typically moves forward from 02:00 to 03:00), any date calculation which would intuitively report a time in that range will also move forward (or backward as the case may be).

NOTE This should be less of a problem since 6.30 with the addition of semi-exact deltas.

Missing date formats

Due to the large number of date formats that Date::Manip CAN process, people often assume that other formats that they want to use should work as well, and when they don’t, it comes as a surprise.

With the much improved parsing of 6.00, many formats can be added easily, though unless they are of general use, I’ll probably suggest that you use parse_format instead.

There is a class of formats that I do not plan to add however.

I have frequently been asked to add formats such as ‘‘the 15th of last month’’, or ‘‘Monday of next week’’. I do not intend to add these date formats to Date::Manip, but since I have received the request several times, I decided to include my reasoning here.

Date::Manip can parse pretty much any static date format that I could think of or find reference to. Dates such as ‘‘today’’, ‘‘Jan 12’’, or ‘‘2001−01−01’’ are all understood.

These are fairly limited however. Many very common date formats are best thought of as a date plus a modification. For example, ‘‘yesterday’’ is actually determined internally as ‘‘today’’ plus a modification of ‘‘− 1 day’’. ‘‘2nd Sunday in June’’ is determined as ‘‘June 1’’ modified to the 2nd Sunday.

As these types of formats were added over time, I quickly realized that the number of possible date plus modification formats was huge. The number of combinations has caused the parsing in Date::Manip to be quite complex, and adding new formats occasionally causes unexpected conflicts with other formats.

The first time I received a request similar to ‘‘the 15th of last month’’, I intended to add it, but as I analyzed it to see what changes needed to be made to support it, I realized that this needed to be expressed as a date plus TWO modifications. In other words, today modified to last month modified to the 15th day of the month.

As bad as date plus modification formats are, a date plus TWO modifications would be exponentially worse. On realizing that, I decided that Date::Manip will not support this type of format.

Although I apologize for the inconvenience, I do not intend to change my position on this.

Date::Manip is slow

Date::Manip is one of the slower Date/Time modules due to the fact that it is huge and written entirely in perl. I have done a lot of work optimizing it since 6.xx came out, and additional work is planned, but even at it’s best, it will probably be slower than other modules.

Some things that will definitely help:

Date::Manip 5.xx was very slow. A lot of work went into speeding it up as I rewrote it for the 6.xx release, and it should be noted that initial tests show version 6.xx to be around twice as fast as 5.xx.

There is one notable exception to this speedup. If you use Date::Manip to parse dates from a wide variety of timezones, 6.xx will be significantly slower than 5.xx. The reason for this is that each time a new timezone is accessed, 6.xx does quite a bit of work to initialize it. 5.xx does not have this overhead, so it can parse dates from any number of timezones without a speed penalty. However, 5.xx does NOT handle timezones correctly, so many of the dates will be incorrect. If timezones are important to you, there is no way to use 5.xx and get accurate results.

If you only parse dates from a single timezone (which is more often what you are doing), 6.xx is significantly faster than 5.xx.

ISO−8601 dates are parsed first and fastest. If you have the flexibility to define the date format, use ISO−8601 formats whenever possible.

Avoid parsing dates that are referenced against the current time (in 2 days, today at noon, etc.). These take a lot longer to parse.

Business date calculations are extremely slow. You should consider alternatives if possible (i.e. doing the calculation in exact mode and then multiplying by 5/7). Who needs a business date more accurate than ‘‘6 to 8 weeks’’ anyway, right :−)

Memory leak

There is a known memory leak in perl related to named regexp captures that directly affects Date::Manip . The leak is in all versions of perl up to (and including) the following versions:

```
5.10.1
5.12.5
5.14.3
5.15.5
```

The bug has been fixed in:

```
5.15.6
5.16.0
```

If a maintenance release is done for any of the other releases (5.10, 5.12, 5.14), that includes the patch, I’ll update this section to include that information.

Date::Manip 5.xx is not susceptible, so using it may be a feasible workaround, but if you need accurate timezone handling, this isn’t possible.

Simple tests estimate the leak to be about 3 MB per 10,000 dates parsed, so unless you’re parsing hundreds of thousands, or millions of dates, the leak probably won’t be a problem on systems with moderate amounts of memory. And if you’re parsing that many dates, the relatively slow Date::Manip may not be the correct module for you to use anyway.

Dmake error on strawberry perl

Users of Strawberry perl on windows may encounter an error similar to the following:

```
dmake: makefile: line 3016: Error: −− Input line too long, increase MAXLINE
```

This is a known problem with some versions of strawberry perl, and I can’t fix it in Date::Manip. If you encounter this problem, you can install the package manually using the commands:

```
c:> cpan
cpan> look Date::Manip::Date
> perl Makefile.PL
> dmake MAXLINELENGTH=300000 make
> dmake MAXLINELENGTH=300000 make test
> dmake MAXLINELENGTH=300000 make install
```

You can find more details here:

```
    http://www.nntp.perl.org/group/perl.win32.vanilla/2011/02/msg287.html
```

Using functions/methods which are not supported

There have been a handful of incidents of people using a function from Date::Manip which were not documented in the manual.

Date::Manip consists of a large number of user functions which are documented in the manual. These are designed to be used by other programmers, and I will not make any backwards incompatible changes in them unless there is a very compelling reason to do so, and in that case, the change will be clearly documented in the Date::Manip::Changes6 documentation for this module.

Date::Manip also includes a large number of functions which are NOT documented. These are for internal use only. Please do not use them! I can (and do) change their functionality, and even their name, without notice, and without apology! Some of these internal functions even have test scripts, but that is not a guarantee that they will not change, nor is any support implied. I simply like to run regression tests on as much of Date::Manip as possible.

As of the most recent versions of Date::Manip, all internal functions have names that begin with an underscore (_). If you choose to use them directly, it is quite possible that new versions of Date::Manip will cause your programs to break due to a change in how those functions work.

Any changes to internal functions will not be documented, and will not be regarded by me as a backwards incompatibility. Nor will I (as was requested in one instance) revert to a previous version of the internal function.

If you feel that an internal function is of more general use, feel free to contact me with an argument of why it should be ‘‘promoted’’. I welcome suggestions and will definitely consider any such request.

RCS Control

If you try to put Date::Manip under RCS control, you are going to have problems. Apparently, RCS replaces strings of the form ‘‘$Date...$’’ with the current date. This form occurs all over in Date::Manip. To prevent the RCS keyword expansion, checkout files using:

```
    co −ko
```

Since very few people will ever have a desire to do this (and I don’t use RCS), I have not worried about it, and I do not intend to try to workaround this problem.

KNOWN COMPLAINTS

Date::Manip 6.xx has gotten some complaints (far more than 5.xx if the truth be told), so I’d like to address a couple of them here. Perhaps an understanding of why some of the changes were made will allay some of the complaints. If not, people are always welcome to stick with the 5.xx release. I will continue to support the 5.xx releases for a couple years (though I do NOT plan to add functionality to it).

These complaints come both from both the CPAN ratings site:

```
    http://cpanratings.perl.org/dist/Date−Manip
```

and from personal email.

Requires perl 5.10

The single most controversial change made in 6.00 is that it now required perl 5.10.0 or higher. Most of the negative feedback I’ve received is due to this.

In the past, I’ve avoided using new features of perl in order to allow Date::Manip to run on older versions of perl. Prior to perl 5.10, none of the new features would have had a major impact on how Date::Manip was written so this practice was justified. That all changed with the release of perl 5.10.

One of the aspects of Date::Manip that has received the most positive response is the ability to parse almost every conceivable date format. Unfortunately, as I’ve added formats, the parsing routine became more and more complicated, and maintaining it was one of the least enjoyable aspect in maintaining Date::Manip . In fact, for several years I’d been extremely reluctant to add new formats due to the fact that too often, adding a new format broke other formats.

As I was rewriting Date::Manip, I was looking for ways to improve the parsing and to make maintaining it easier. Perl 5.10 provides the feature ''named capture buffers''. Named capture buffers not only improves the ease of maintaining the complex regular expressions used by Date::Manip, it makes it dramatically easier to add additional formats in a way that is much less likely to interfere with other formats. The parsing in 6.00 is so much more robust, extensible, and flexible, that it will make parser maintenance possible for many years to come at a fraction of the effort and risk.

It was too much to turn down. Hopefully, since 5.10 has been out for some time now, this will not prohibit too many people from using the new version of Date::Manip. I realize that there are many people out there using older versions of perl who do not have the option of upgrading perl. The decision to use 5.10 wasn't made lightly... but I don't regret making it. I apologize to users who, as a result, cannot use 6.00 . Hopefully in the future you'll be able to benefit from the improvements in 6.00.

One complaint I've received is that this in some way makes Date::Manip backwards incompatible, but this is not an accurate complaint. Version 6.xx DOES include some backwards incompatibilities (and these are covered in the Date::Manip::Migration5to6 document), however in almost all cases, these incompatibilities are with infrequently used features, or workarounds are in place to allow deprecated features to continue functioning for some period of time.

Though I have no data to confirm this, I suspect that 90% or more of all scripts which were written with Date::Manip 5.xx will continue to work unmodified with 6.xx (of course, you should still refer to the migration document to see what features are deprecated or changed to make sure that you don't need to modify your script so that it will continue to work in the future). Even with scripts that need to be changed, the changes should be trivial.

So, Date::Manip 6.xx is almost entirely backward compatible with 5.xx (to the extent that you would expect any major version release to be compatible with a previous major version).

The change is only in the requirements necessary to get Date::Manip 6.xx to run.

Obviously, it's not reasonable to say that Date::Manip should never be allowed to switch minimum perl versions. At some point, you have to let go of an old version if you want to make use of the features of the newer version. The question is, did I jump to 5.10 too fast?

The negative ratings I see in the CPAN ratings complain that I no longer support perl 5.6 and perl 5.8.

With respect to 5.6, perl 5.6 was released in March of 2000 (that's before Windows XP which was released in 2001). Date::Manip 6.00 was released at the end of 2009. To be honest, I don't really feel much sympathy for this complaint. Software that is 9 years old is ANCIENT. People may choose to use it... but please don't complain when new software comes out that doesn't support it.

The argument for perl 5.8 is much more compelling. Although 5.8 was released well before Date::Manip 6.00 (July of 2002), there were no major perl releases until 5.10 came out in December of 2007, so 5.8 was state-of-the art as little as 2 years prior to the release of Date::Manip 6.xx.

I agree completely with the argument that abandoning 5.8 only 2 years after it was the current version is too soon. For that reason, I continued to support the Date::Manip 5.xx releases for several years. As of December 2012 (5 years after the release of perl 5.10), the 5.xx release is no longer supported (in that I will not patch it or provide support for it's use). However, it is still bundled into the Date::Manip distribution so it can still be used. I do not have any plans for removing it, though I may do so at some point.

Too many modules

One minor complaint is that there are too many files. One person specifically objects to the fact that there are over 470 modules covering non-minute offsets. This complaint is (IMO) silly.

Date::Manip supports ALL historical time zones, including those with non-minute offsets, and so there will be information for those time zones, even though they are not currently in use.

I could of course store all of the information in one big module, but that means that you have to load all of that data every time you use Date::Manip, and I find that to be a very poor solution. Instead, storing the information in a per-time zone and per-offset manner dramatically decreases the amount of data that has to be loaded.

While it is true that Date::Manip includes over 900 modules for all of the time zone information, most implementations of time zone handling also choose to break up the data into a large number of files.

My linux distribution (openSuSE 11.2 at the time of writing this) uses the standard zoneinfo database, and at this point, there are over 1700 files included in /usr/share/zoneinfo (though it does appear that there is some duplication of information). Current versions of RedHat also use over 1700 files, so Date::Manip isn't treating the time zone data in a new or unreasonable way.

Objects are large

One complaint that was put on the CPAN ratings site was that the OO interface is ‘‘a dud’’ due to the size of it's objects. The complaint is that a Date::Manip::Date object is 115K while it should (according to the complaint) only require that you need to save the seconds from the epoch, zone, and a couple other pieces of information, all of which could probably be stored in 100 bytes or less.

This complaint is not accurate, and comes from a misunderstanding of the objects used by Date::Manip.

Date::Manip is very configurable, and contains a great deal of information which could theoretically be calculated on the fly, but which would greatly reduce it's performance. Instead, in the interest of improving performance, the data is cached, and since the data is virtually all potentially object specific, it has to be somehow linked to the object.

For example, Date::Manip allows you to parse dates in several languages. Each language has a large number of regular expressions which are used to do the actual parsing. Instead of recreating these regular expressions each time they are needed, they are created once and stored in an object (specifically, a Date::Manip::Base object). The size of the Date::Manip::Base object is almost 15K (due primarily to the regular expressions used in parsing dates in the selected language).

Similarly, a description of the time zones are stored in a second object (a Date::Manip::TZ object). The size of the Date::Manip::TZ object starts at 100K. That may seem excessive, but you have to remember that there are almost 500 time zones, and they have to be indexed by name, alias, abbreviation, and offset. In addition, critical dates (dates where the offset from GMT change such as during a daylight saving time change) along with information such as offsets, abbreviation, etc., are all cached in order to improve performance. By the time you do this, it does take a fair bit of space. It should also be noted that the full description of each timezone is only stored in the object when the timezone is actually used, so if you use a lot of timezones, this object will grow slowly as new timezones are used.

The size of the actual Date::Manip::Date object is a little over 300 bytes. However, each includes a pointer to a Date::Manip::Base and a Date::Manip::TZ object (and due to how the object was being looked at in the complaint, they were reporting the size of all three objects, NOT just the Date::Manip::Date object).

Both the Date::Manip::Base and Date::Manip::TZ objects are reused by any number of Date::Manip::Date objects. They can almost be thought of as global data, except that they are accessible in the standard OO manner, and you are allowed to modify them on a per-object basis which WILL mean that you have to store more data. If you work with multiple configurations (see Date::Manip::Config), you'll need multiple Base and TZ objects. However, most of the time you will not need to do this.

The actual Date::Manip::Date object is a bit larger than suggested in the complaint, but it should be noted that Date::Manip actually stores the dates in a number of different formats (a string of the form YYYYMMDDHH:MN:SS and a list [YYYY,MM,DD,HH,MN,SS] in the time zone it was parsed in, the local time zone (if different) and GMT. By caching this information as it is used, it has a huge impact on the performance.

So, Date::Manip in typical usage consists of one 100K Date::Manip::TZ object, one 15K Date::Manip::Base objects, and any number of small 300 byte Date::Manip::Date objects. Date::Manip::Delta objects are even smaller. Date::Manip::Recur objects are also small, but they contain any number of Date objects in them.

I am certainly open to suggestions as to how I can improve the OO interface... but I don't believe it is a dud. While I'm not an expert at OO programming, I believe that I followed pretty standard and accepted procedures for accessing the data.

Please refer to the Date::Manip::Objects document for more information.

Date::Manip has an inconsistent interface

I've gotten a few complaints that the interface to Date::Manip is inconsistent... and I agree (at least when referring to the functional interfaces).

Date::Manip was originally written in an unplanned way... as a need/want came up, it was extended. That's not the way to write a major package of course, but it wasn't expected to be a major package at the start.

As it became more and more widely used, I too wished for a more consistent interface, but I did not want to break backward compatibility for everyone using it.

When 6.xx was written, I spent a good deal of time trying to make a very standard OO interface, so I do not believe that this complaint can be applied to the OO interface (though I'm interested in suggestions for improving it of course).

As far as the functional interface goes, I'll continue to support it in a backward compatible (and therefore inconsistent) form. I'd encourage the use of the OO interface whenever possible.

BUGS AND QUESTIONS

If you find a bug in Date::Manip, there are three ways to send it to me. In order of preference, they are:

GitHub

You can submit it as an issue on GitHub. This can be done at the following URL:

<https://github.com/SBECK−github/Date−Manip>

This is the preferred method. Please submit problems requests as GitHub issues if at all possible.

Direct email

You are welcome to send it directly to me by email. The email address to use is: sbeck@cpan.org.

Please note that because cpan.org addresses are published, they are used by a lot of spammers and phishers. Please include the name of the perl module in the subject line of ALL messages sent to my cpan.org address or it will likely be missed.

CPAN Bug Tracking

You can submit it using the CPAN tracking too. This can be done at the following URL:

<http://rt.cpan.org/Public/Dist/Display.html?Name=Date−Manip>

There was discussion of halting this service a while back (though it continues to function), so only use this as a last resort.

Please do not use other means to report bugs (such as forums for a specific OS or Linux distribution) as it is impossible for me to keep up with all of them.

When filing a bug report, please include the following information:

Date::Manip version

Please include the version of Date::Manip you are using. You can get this by using the script:

```
use Date::Manip;
print DateManipVersion(1),"\n";
```

or

```
use Date::Manip::Date;
$obj = new Date::Manip::Date;
print $obj->version(1),"\n";
```

Perl information

Please include the output from ‘‘perl −V’’

If you have a problem using Date::Manip that perhaps isn’t a bug (can’t figure out the syntax, etc.), you’re in the right place. Start by reading the main Date::Manip documentation, and the other documents that apply to whatever you are trying to do. If this still doesn’t answer your question, mail me directly.

I would ask that you be reasonably familiar with the documentation BEFORE you choose to do this. Date::Manip is a hobby, and I simply do not have time to respond to hundreds of questions which are already answered in this manual.

If you find any problems with the documentation (errors, typos, or items that are not clear), please send them to me. I welcome any suggestions that will allow me to improve the documentation.

KNOWN BUGS

None known.

SEE ALSO

Date::Manip	− main module documentation

LICENSE

AUTHOR

Sullivan Beck (sbeck@cpan.org)