NAME

perl591delta - what is new for perl v5.9.1

DESCRIPTION

This document describes differences between the 5.9.0 and the 5.9.1 development releases. See *perl590delta* for the differences between 5.8.0 and 5.9.0.

Incompatible Changes

substr() lvalues are no longer fixed-length

The lvalues returned by the three argument form of substr() used to be a "fixed length window" on the original string. In some cases this could cause surprising action at distance or other undefined behaviour. Now the length of the window adjusts itself to the length of the string assigned to it.

The :unique attribute is only meaningful for globals

Now applying `:unique` to lexical variables and to subroutines will result in a compilation error.

Core Enhancements

Lexical $_

The default variable $_ can now be lexicalized, by declaring it like any other lexical variable, with a simple

```
my $_;
```

The operations that default on $_ will use the lexically-scoped version of $_ when it exists, instead of the global $_.

In a `map` or a `grep` block, if $_ was previously my'ed, then the $_ inside the block is lexical as well (and scoped to the block).

In a scope where $_ has been lexicalized, you can still have access to the global version of $_ by using `$::_`, or, more simply, by overriding the lexical declaration with `our $_`.

Tied hashes in scalar context

As of perl 5.8.2/5.9.0, tied hashes did not return anything useful in scalar context, for example when used as boolean tests:

```
if (%tied_hash) { ... }
```

The old nonsensical behaviour was always to return false, regardless of whether the hash is empty or has elements.

There is now an interface for the implementors of tied hashes to implement the behaviour of a hash in scalar context, via the SCALAR method (see *perltie*). Without a SCALAR method, perl will try to guess whether the hash is empty, by testing if it's inside an iteration (in this case it can't be empty) or by calling FIRSTKEY.

Formats

Formats were improved in several ways. A new field, `^*`, can be used for variable-width, one-line-at-a-time text. Null characters are now handled correctly in picture lines. Using @# and ~~ together will now produce a compile-time error, as those format fields are incompatible. *perlform* has been improved, and miscellaneous bugs fixed.

Stacked filetest operators

As a new form of syntactic sugar, it's now possible to stack up filetest operators. You can now write `-f -w -x $file` in a row to mean `-x $file && -w _ && -f _`. See *"-X" in perlfunc*.

Modules and Pragmata

Benchmark

In `Benchmark`, cmpthese() and timestr() now use the time statistics of children instead of parent when the selected style is 'nop'.

Carp

The error messages produced by `Carp` now include spaces between the arguments in function argument lists: this makes long error messages appear more nicely in browsers and other tools.

Exporter

`Exporter` will now recognize grouping tags (such as `:name`) anywhere in the import list, not only at the beginning.

FindBin

A function `again` is provided to resolve problems where modules in different directories wish to use FindBin.

List::Util

You can now weaken references to read only values.

threads::shared

`cond_wait` has a new two argument form. `cond_timedwait` has been added.

Utility Changes

`find2perl` now assumes `-print` as a default action. Previously, it needed to be specified explicitly.

A new utility, `prove`, makes it easy to run an individual regression test at the command line. `prove` is part of Test::Harness, which users of earlier Perl versions can install from CPAN.

The perl debugger now supports a `save` command, to save the current history to a file, and an `i` command, which prints the inheritance tree of its argument (if the `Class::ISA` module is installed.)

Documentation

The documentation has been revised in places to produce more standard manpages.

The long-existing feature of `/(?{...})/` regexps setting `$_` and pos() is now documented.

Performance Enhancements

Sorting arrays in place (`@a = sort @a`) is now optimized to avoid making a temporary copy of the array.

The operations involving case mapping on UTF-8 strings (uc(), lc(), `//i`, etc.) have been greatly speeded up.

Access to elements of lexical arrays via a numeric constant between 0 and 255 is now faster. (This used to be only the case for global arrays.)

Selected Bug Fixes

UTF-8 bugs

Using substr() on a UTF-8 string could cause subsequent accesses on that string to return garbage. This was due to incorrect UTF-8 offsets being cached, and is now fixed.

join() could return garbage when the same join() statement was used to process 8 bit data having earlier processed UTF-8 data, due to the flags on that statement's temporary workspace not being reset correctly. This is now fixed.

Using Unicode keys with tied hashes should now work correctly.

chop() and chomp() used to mangle UTF-8 strings. This has been fixed.

sprintf() used to misbehave when the format string was in UTF-8. This is now fixed.

Threading bugs

Hashes with the `:unique` attribute weren't made read-only in new threads. They are now.

More bugs

`$a .. $b` will now work as expected when either $a or $b is `undef`.

Reading $^E now preserves $!. Previously, the C code implementing $^E did not preserve `errno`, so reading $^E could cause `errno` and therefore `$!` to change unexpectedly.

`strict` wasn't in effect in regexp-eval blocks (`/(?{...})/`).

New or Changed Diagnostics

A new deprecation warning, *Deprecated use of my() in false conditional*, has been added, to warn against the use of the dubious and deprecated construct

```
my $x if 0;
```

See *perldiag*.

The fatal error *DESTROY created new reference to dead object* is now documented in *perldiag*.

A new error, *%ENV is aliased to %s*, is produced when taint checks are enabled and when `*ENV` has been aliased (and thus doesn't reflect the program's environment anymore.)

Changed Internals

These news matter to you only if you either write XS code or like to know about or hack Perl internals (using Devel::Peek or any of the `B::` modules counts), or like to run Perl with the `-D` option.

Reordering of SVt_* constants

The relative ordering of constants that define the various types of SV have changed; in particular, `SVt_PVGV` has been moved before `SVt_PVLV`, `SVt_PVAV`, `SVt_PVHV` and `SVt_PVCV`. This is unlikely to make any difference unless you have code that explicitly makes assumptions about that ordering. (The inheritance hierarchy of `B::*` objects has been changed to reflect this.)

Removal of CPP symbols

The C preprocessor symbols `PERL_PM_APIVERSION` and `PERL_XS_APIVERSION`, which were supposed to give the version number of the oldest perl binary-compatible (resp. source-compatible) with the present one, were not used, and sometimes had misleading values. They have been removed.

Less space is used by ops

The `BASEOP` structure now uses less space. The `op_seq` field has been removed and replaced by two one-bit fields, `op_opt` and `op_static`. `opt_type` is now 9 bits long. (Consequently, the `B::OP` class doesn't provide an `seq` method anymore.)

New parser

perl's parser is now generated by bison (it used to be generated by byacc.) As a result, it seems to be a bit more robust.

Configuration and Building

`Configure` now invokes callbacks regardless of the value of the variable they are called for. Previously callbacks were only invoked in the `case $variable $define)` branch. This change

should only affect platform maintainers writing configuration hints files.

The portability and cleanliness of the Win32 makefiles has been improved.

Known Problems

There are still a couple of problems in the implementation of the lexical $_: it doesn't work inside `/(?{...})/` blocks and with regard to the reverse() built-in used without arguments. (See the TODO tests in *t/op/mydef.t*.)

Platform Specific Problems

The test *ext/IPC/SysV/t/ipcsysv.t* may fail on OpenBSD. This hasn't been diagnosed yet.

On some configurations on AIX 5, one test in *lib/Time/Local.t* fails. When configured with long doubles, perl may fail tests 224-236 in *t/op/pow.t* on the same platform.

For threaded builds, *ext/threads/shared/t/wait.t* has been reported to fail some tests on HP-UX 10.20.

To-do for perl 5.10.0

This is a non-exhaustive, non-ordered, non-contractual and non-definitive list of things to do (or nice to have) for perl 5.10.0 :

Clean up and finish support for assertions. See *assertions*.

Reimplement the mechanism of lexical pragmas to be more extensible. Fix current pragmas that don't work well (or at all) with lexical scopes or in run-time eval(STRING) (`sort`, `re`, `encoding` for example). MJD has a preliminary patch that implements this.

Fix (or rewrite) the implementation of the `/(?{...})/` closures.

Conversions from byte strings to UTF-8 currently map high bit characters to Unicode without translation (or, depending on how you look at it, by implicitly assuming that the byte strings are in Latin-1). As perl assumes the C locale by default, upgrading a string to UTF-8 may change the meaning of its contents regarding character classes, case mapping, etc. This should probably emit a warning (at least).

Introduce a new special block, UNITCHECK, which is run at the end of a compilation unit (module, file, eval(STRING) block). This will correspond to the Perl 6 CHECK. Perl 5's CHECK cannot be changed or removed because the O.pm/B.pm backend framework depends on it.

Study the possibility of adding a new prototype character, _, meaning "this argument defaults to $_".

Make the peephole optimizer optional.

Allow lexical aliases (maybe via the syntax `my \$alias = \$foo`.

Fix the bugs revealed by running the test suite with the `-t` switch (via `make test.taintwarn`).

Make threads more robust.

Make `no 6` and `no v6` work (opposite of `use 5.005`, etc.).

A test suite for the B module would be nice.

A ponie.

Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with

the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.