NAME

perl595delta - what is new for perl v5.9.5

DESCRIPTION

This document describes differences between the 5.9.4 and the 5.9.5 development releases. See *perl590delta*, *perl591delta*, *perl592delta*, *perl593delta* and *perl594delta* for the differences between 5.8.0 and 5.9.4.

Incompatible Changes

Tainting and printf

When perl is run under taint mode, `printf()` and `sprintf()` will now reject any tainted format argument. (Rafael Garcia-Suarez)

undef and signal handlers

Undefining or deleting a signal handler via `undef $SIG{FOO}` is now equivalent to setting it to `'DEFAULT'`. (Rafael)

strictures and array/hash dereferencing in defined()

`defined @$foo` and `defined %$bar` are now subject to `strict 'refs'` (that is, `$foo` and `$bar` shall be proper references there.) (Nicholas Clark)

(However, `defined(@foo)` and `defined(%bar)` are discouraged constructs anyway.)

(?p{}) has been removed

The regular expression construct `(?p{})`, which was deprecated in perl 5.8, has been removed. Use `(??{})` instead. (Rafael)

Pseudo-hashes have been removed

Support for pseudo-hashes has been removed from Perl 5.9. (The `fields` pragma remains here, but uses an alternate implementation.)

Removal of the bytecode compiler and of perlcc

`perlcc`, the byteloader and the supporting modules (B::C, B::CC, B::Bytecode, etc.) are no longer distributed with the perl sources. Those experimental tools have never worked reliably, and, due to the lack of volunteers to keep them in line with the perl interpreter developments, it was decided to remove them instead of shipping a broken version of those. The last version of those modules can be found with perl 5.9.4.

However the B compiler framework stays supported in the perl core, as with the more useful modules it has permitted (among others, B::Deparse and B::Concise).

Removal of the JPL

The JPL (Java-Perl Linguo) has been removed from the perl sources tarball.

Recursive inheritance detected earlier

Perl will now immediately throw an exception if you modify any package's `@ISA` in such a way that it would cause recursive inheritance.

Previously, the exception would not occur until Perl attempted to make use of the recursive inheritance while resolving a method or doing a `$foo->isa($bar)` lookup.

Core Enhancements

Regular expressions

Recursive Patterns

It is now possible to write recursive patterns without using the `(??{})` construct. This new way is more efficient, and in many cases easier to read.

Each capturing parenthesis can now be treated as an independent pattern that can be entered by using the `(?PARNO)` syntax (`PARNO` standing for "parenthesis number"). For example, the following pattern will match nested balanced angle brackets:

```
    /
     ^                     # start of line
     (                     # start capture buffer 1
 <                   #   match an opening angle bracket
 (?:                 #   match one of:
     (?>             #     don't backtrack over the inside of this
group
  [^<>]+      #       one or more non angle brackets
     )               #     end non backtracking group
 |                   #     ... or ...
     (?1)            #     recurse to bracket 1 and try it again
 )*                  #   0 or more times.
 >                   #   match a closing angle bracket
     )                     # end capture buffer one
     $                     # end of line
    /x
```

Note, users experienced with PCRE will find that the Perl implementation of this feature differs from the PCRE one in that it is possible to backtrack into a recursed pattern, whereas in PCRE the recursion is atomic or "possessive" in nature. (Yves Orton)

Named Capture Buffers

It is now possible to name capturing parenthesis in a pattern and refer to the captured contents by name. The naming syntax is `(?<NAME>....)`. It's possible to backreference to a named buffer with the `\k<NAME>` syntax. In code, the new magical hashes `%+` and `%-` can be used to access the contents of the capture buffers.

Thus, to replace all doubled chars, one could write

```
    s/(?<letter>.)\k<letter>/$+{letter}/g
```

Only buffers with defined contents will be "visible" in the `%+` hash, so it's possible to do something like

```
    foreach my $name (keys %+) {
        print "content of buffer '$name' is $+{$name}\n";
    }
```

The `%-` hash is a bit more complete, since it will contain array refs holding values from all capture buffers similarly named, if there should be many of them.

`%+` and `%-` are implemented as tied hashes through the new module `Tie::Hash::NamedCapture`.

Users exposed to the .NET regex engine will find that the perl implementation differs in that the numerical ordering of the buffers is sequential, and not "unnamed first, then named". Thus in the pattern

```
   /(A)(?<B>B)(C)(?<D>D)/
```

$1 will be 'A', $2 will be 'B', $3 will be 'C' and $4 will be 'D' and not $1 is 'A', $2 is 'C' and $3 is 'B' and $4 is 'D' that a .NET programmer would expect. This is considered a feature. :-) (Yves Orton)

Possessive Quantifiers

Perl now supports the "possessive quantifier" syntax of the "atomic match" pattern. Basically a possessive quantifier matches as much as it can and never gives any back. Thus it can be

used to control backtracking. The syntax is similar to non-greedy matching, except instead of using a '?' as the modifier the '+' is used. Thus `?+`, `*+`, `++`, `{min,max}+` are now legal quantifiers. (Yves Orton)

Backtracking control verbs

The regex engine now supports a number of special-purpose backtrack control verbs: (*THEN), (*PRUNE), (*MARK), (*SKIP), (*COMMIT), (*FAIL) and (*ACCEPT). See *perlre* for their descriptions. (Yves Orton)

Relative backreferences

A new syntax `\g{N}` or `\gN` where "N" is a decimal integer allows a safer form of back-reference notation as well as allowing relative backreferences. This should make it easier to generate and embed patterns that contain backreferences. See *"Capture buffers" in perlre*. (Yves Orton)

`\K` escape

The functionality of Jeff Pinyan's module Regexp::Keep has been added to the core. You can now use in regular expressions the special escape `\K` as a way to do something like floating length positive lookbehind. It is also useful in substitutions like:

```
s/(foo)bar/$1/g
```

that can now be converted to

```
s/foo\Kbar//g
```

which is much more efficient. (Yves Orton)

Vertical and horizontal whitespace, and linebreak

Regular expressions now recognize the `\v` and `\h` escapes, that match vertical and horizontal whitespace, respectively. `\V` and `\H` logically match their complements.

`\R` matches a generic linebreak, that is, vertical whitespace, plus the multi-character sequence `"\x0D\x0A"`.

The _ prototype

A new prototype character has been added. _ is equivalent to $ (it denotes a scalar), but defaults to $_ if the corresponding argument isn't supplied. Due to the optional nature of the argument, you can only use it at the end of a prototype, or before a semicolon.

This has a small incompatible consequence: the prototype() function has been adjusted to return _ for some built-ins in appropriate cases (for example, `prototype('CORE::rmdir')`). (Rafael)

UNITCHECK blocks

`UNITCHECK`, a new special code block has been introduced, in addition to `BEGIN`, `CHECK`, `INIT` and `END`.

`CHECK` and `INIT` blocks, while useful for some specialized purposes, are always executed at the transition between the compilation and the execution of the main program, and thus are useless whenever code is loaded at runtime. On the other hand, `UNITCHECK` blocks are executed just after the unit which defined them has been compiled. See *perlmod* for more information. (Alex Gough)

readpipe() is now overridable

The built-in function readpipe() is now overridable. Overriding it permits also to override its operator counterpart, `qx//` (a.k.a. ``` `` ```). Moreover, it now defaults to $_ if no argument is provided. (Rafael)

default argument for readline()

readline() now defaults to `*ARGV` if no argument is provided. (Rafael)

UCD 5.0.0

The copy of the Unicode Character Database included in Perl 5.9 has been updated to version 5.0.0.

Smart match

The smart match operator (~~) is now available by default (you don't need to enable it with `use feature` any longer). (Michael G Schwern)

Implicit loading of feature

The `feature` pragma is now implicitly loaded when you require a minimal perl version (with the `use VERSION` construct) greater than, or equal to, 5.9.5.

Modules and Pragmas

New Pragma, mro

A new pragma, `mro` (for Method Resolution Order) has been added. It permits to switch, on a per-class basis, the algorithm that perl uses to find inherited methods in case of a multiple inheritance hierarchy. The default MRO hasn't changed (DFS, for Depth First Search). Another MRO is available: the C3 algorithm. See *mro* for more information. (Brandon Black)

Note that, due to changes in the implementation of class hierarchy search, code that used to undef the `*ISA` glob will most probably break. Anyway, undef'ing `*ISA` had the side-effect of removing the magic on the @ISA array and should not have been done in the first place.

bignum, bigint, bigrat

The three numeric pragmas `bignum`, `bigint` and `bigrat` are now lexically scoped. (Tels)

Math::BigInt/Math::BigFloat

Many bugs have been fixed; noteworthy are comparisons with NaN, which no longer warn about undef values.

The following things are new:

config()

The config() method now also supports the calling-style `config('lib')` in addition to `config()->{'lib'}`.

import()

Upon import, using `lib => 'Foo'` now warns if the low-level library cannot be found. To suppress the warning, you can use `try => 'Foo'` instead. To convert the warning into a die, use `only => 'Foo'` instead.

roundmode common

A rounding mode of `common` is now supported.

Also, support for the following methods has been added:

bpi(), bcos(), bsin(), batan(), batan2()

bmuladd()

bexp(), bnok()

from_hex(), from_oct(), and from_bin()

as_oct()

In addition, the default math-backend (Calc (Perl) and FastCalc (XS)) now support storing numbers in parts with 9 digits instead of 7 on Perls with either 64bit integer or long double support. This means math operations scale better and are thus faster for really big numbers.

New Core Modules

- `Locale::Maketext::Simple`, needed by CPANPLUS, is a simple wrapper around `Locale::Maketext::Lexicon`. Note that `Locale::Maketext::Lexicon` isn't included in the perl core; the behaviour of `Locale::Maketext::Simple` gracefully degrades when the later isn't present.
- `Params::Check` implements a generic input parsing/checking mechanism. It is used by CPANPLUS.
- `Term::UI` simplifies the task to ask questions at a terminal prompt.
- `Object::Accessor` provides an interface to create per-object accessors.
- `Module::Pluggable` is a simple framework to create modules that accept pluggable sub-modules.
- `Module::Load::Conditional` provides simple ways to query and possibly load installed modules.
- `Time::Piece` provides an object oriented interface to time functions, overriding the built-ins localtime() and gmtime().
- `IPC::Cmd` helps to find and run external commands, possibly interactively.
- `File::Fetch` provide a simple generic file fetching mechanism.
- `Log::Message` and `Log::Message::Simple` are used by the log facility of `CPANPLUS`.
- `Archive::Extract` is a generic archive extraction mechanism for *.tar* (plain, gziped or bzipped) or *.zip* files.
- `CPANPLUS` provides an API and a command-line tool to access the CPAN mirrors.

Module changes

`assertions`

The `assertions` pragma, its submodules `assertions::activate` and `assertions::compat` and the **-A** command-line switch have been removed. The interface was not judged mature enough for inclusion in a stable release.

`base`

The `base` pragma now warns if a class tries to inherit from itself. (Curtis "Ovid" Poe)

`strict` and `warnings`

`strict` and `warnings` will now complain loudly if they are loaded via incorrect casing (as in `use Strict;`). (Johan Vromans)

`warnings`

The `warnings` pragma doesn't load `Carp` anymore. That means that code that used `Carp` routines without having loaded it at compile time might need to be adjusted; typically, the following (faulty) code won't work anymore, and will require parentheses to be added after the function name:

```
use warnings;
require Carp;
Carp::confess "argh";
```

`less`

`less` now does something useful (or at least it tries to). In fact, it has been turned into a lexical pragma. So, in your modules, you can now test whether your users have requested to

use less CPU, or less memory, less magic, or maybe even less fat. See *less* for more. (Joshua ben Jore)

`Attribute::Handlers`

`Attribute::Handlers` can now report the caller's file and line number. (David Feldman)

`B::Lint`

`B::Lint` is now based on `Module::Pluggable`, and so can be extended with plugins. (Joshua ben Jore)

`B`

It's now possible to access the lexical pragma hints (`%^H`) by using the method B::COP::hints_hash(). It returns a `B::RHE` object, which in turn can be used to get a hash reference via the method B::RHE::HASH(). (Joshua ben Jore)

`Thread`

As the old 5005thread threading model has been removed, in favor of the ithreads scheme, the `Thread` module is now a compatibility wrapper, to be used in old code only. It has been removed from the default list of dynamic extensions.

Utility Changes

cpanp

`cpanp`, the CPANPLUS shell, has been added. (`cpanp-run-perl`, an helper for CPANPLUS operation, has been added too, but isn't intended for direct use).

cpan2dist

`cpan2dist` is a new utility, that comes with CPANPLUS. It's a tool to create distributions (or packages) from CPAN modules.

pod2html

The output of `pod2html` has been enhanced to be more customizable via CSS. Some formatting problems were also corrected. (Jari Aalto)

Documentation

New manpage, perlunifaq

A new manual page, *perlunifaq* (the Perl Unicode FAQ), has been added (Juerd Waalboer).

Installation and Configuration Improvements

C++ compatibility

Efforts have been made to make perl and the core XS modules compilable with various C++ compilers (although the situation is not perfect with some of the compilers on some of the platforms tested.)

Visual C++

Perl now can be compiled with Microsoft Visual C++ 2005.

Static build on Win32

It's now possible to build a `perl-static.exe` that doesn't depend on `perl59.dll` on Win32. See the Win32 makefiles for details. (Vadim Konovalov)

win32 builds

All win32 builds (MS-Win, WinCE) have been merged and cleaned up.

d_pseudofork and d_printf_format_null

A new configuration variable, available as `$Config{d_pseudofork}` in the *Config* module, has been added, to distinguish real fork() support from fake pseudofork used on Windows platforms.

A new configuration variable, `d_printf_format_null`, has been added, to see if printf-like formats are allowed to be NULL.

Help

`Configure -h` has been extended with the most used option.

Much less 'Whoa there' messages.

64bit systems

Better detection of 64bit(only) systems, and setting all the (library) paths accordingly.

Ports

Perl has been reported to work on MidnightBSD.

Support for Cray XT4 Catamount/Qk has been added.

Vendor patches have been merged for RedHat and GenToo.

Selected Bug Fixes

PerlIO::scalar will now prevent writing to read-only scalars. Moreover, seek() is now supported with PerlIO::scalar-based filehandles, the underlying string being zero-filled as needed. (Rafael, Jarkko Hietaniemi)

study() never worked for UTF-8 strings, but could lead to false results. It's now a no-op on UTF-8 data. (Yves Orton)

The signals SIGILL, SIGBUS and SIGSEGV are now always delivered in an "unsafe" manner (contrary to other signals, that are deferred until the perl interpreter reaches a reasonably stable state; see *"Deferred Signals (Safe Signals)" in perlipc*). (Rafael)

When a module or a file is loaded through an @INC-hook, and when this hook has set a filename entry in %INC, __FILE__ is now set for this module accordingly to the contents of that %INC entry. (Rafael)

The `-w` and `-t` switches can now be used together without messing up what categories of warnings are activated or not. (Rafael)

Duping a filehandle which has the `:utf8` PerlIO layer set will now properly carry that layer on the duped filehandle. (Rafael)

Localizing an hash element whose key was given as a variable didn't work correctly if the variable was changed while the local() was in effect (as in `local $h{$x}; ++$x`). (Bo Lindbergh)

New or Changed Diagnostics

Deprecations

Two deprecation warnings have been added: (Rafael)

```
Opening dirhandle %s also as a file
Opening filehandle %s also as a directory
```

Changed Internals

The anonymous hash and array constructors now take 1 op in the optree instead of 3, now that pp_anonhash and pp_anonlist return a reference to an hash/array when the op is flagged with OPf_SPECIAL (Nicholas Clark).

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://rt.perl.org/rt3/ . 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.