NAME

perl5216delta - what is new for perl v5.21.6

DESCRIPTION

This document describes differences between the 5.21.5 release and the 5.21.6 release.

If you are upgrading from an earlier release such as 5.21.4, first read *perl5215delta*, which describes differences between 5.21.4 and 5.21.5.

Core Enhancements

List form of pipe open implemented for Win32

The list form of pipe:

```
open my $fh, "-|", "program", @arguments;
```

is now implemented on Win32. It has the same limitations as `system LIST` on Win32, since the Win32 API doesn't accept program arguments as a list.

Assignment to list repetition

`(...) x ...` can now be used within a list that is assigned to, as long as the left-hand side is a valid lvalue. This allows `(undef,undef,$foo) = that_function()` to be written as `((undef)x2, $foo) = that_function()`.

close now sets $!

When an I/O error occurs, the fact that there has been an error is recorded in the handle. `close` returns false for such a handle. Previously, the value of `$!` would be untouched by `close`, so the common convention of writing `close $fh or die $!` did not work reliably. Now the handle records the value of `$!`, too, and `close` restores it.

Deprecations

Use of non-graphic characters in single-character variable names

The syntax for single-character variable names is more lenient than for longer variable names, allowing the one-character name to be a punctuation character or even invisible (a non-graphic). Perl v5.20 deprecated the ASCII-range controls as such a name. Now, all non-graphic characters that formerly were allowed are deprecated. The practical effect of this occurs only when not under `"use utf8"`, and affects just the C1 controls (code points 0x80 through 0xFF), NO-BREAK SPACE, and SOFT HYPHEN.

Inlining of sub () { $var } with observable side-effects

In many cases Perl makes sub () { $var } into an inlinable constant subroutine, capturing the value of $var at the time the `sub` expression is evaluated. This can break the closure behaviour in those cases where $var is subsequently modified. The subroutine won't return the new value.

This usage is now deprecated in those cases where the variable could be modified elsewhere. Perl detects those cases and emits a deprecation warning. Such code will likely change in the future and stop producing a constant.

If your variable is only modified in the place where it is declared, then Perl will continue to make the sub inlinable with no warnings.

```
sub make_constant {
    my $var = shift;
    return sub () { $var }; # fine
}

sub make_constant_deprecated {
```

```
    my $var;
    $var = shift;
    return sub () { $var }; # deprecated
}

sub make_constant_deprecated2 {
    my $var = shift;
    log_that_value($var); # could modify $var
    return sub () { $var }; # deprecated
}
```

In the second example above, detecting that $var is assigned to only once is too hard to detect. That it happens in a spot other than the `my` declaration is enough for Perl to find it suspicious.

This deprecation warning happens only for a simple variable for the body of the sub. (A `BEGIN` block or `use` statement inside the sub is ignored, because it does not become part of the sub's body.) For more complex cases, such as `sub () { do_something() if 0; $var }` the behaviour has changed such that inlining does not happen if the variable is modifiable elsewhere. Such cases should be rare.

Performance Enhancements

- `(...)x1`, `("constant")x0` and `($scalar)x0` are now optimised in list context. If the right-hand argument is a constant 1, the repetition operator disappears. If the right-hand argument is a constant 0, the whole expressions is optimised to the empty list, so long as the left-hand argument is a simple scalar or constant. `(foo())x0` is not optimised.
- `substr` assignment is now optimised into 4-argument `substr` at the end of a subroutine (or as the argument to `return`). Previously, this optimisation only happened in void context.
- Assignment to lexical variables is often optimised away. For instance, in `$lexical = chr $foo`, the `chr` operator writes directly to the lexical variable instead of returning a value that gets copied. This optimisation has been extended to `split`, `x` and `vec` on the right-hand side. It has also been made to work with state variable initialization.
- In "\L...", "\Q...", etc., the extra "stringify" op is now optimised away, making these just as fast as `lcfirst`, `quotemeta`, etc.
- Assignment to an empty list is now sometimes faster. In particular, it never calls `FETCH` on tied arguments on the right-hand side, whereas it used to sometimes.

Modules and Pragmata

Updated Modules and Pragmata

- *B* has been upgraded from version 1.52 to 1.53.
- *B::Concise* has been upgraded from version 0.994 to 0.995.
- *B::Deparse* has been upgraded from version 1.29 to 1.30.

  It now deparses `+sub : attr { ... }` correctly at the start of a statement. Without the initial +, `sub` would be a statement label.

  `BEGIN` blocks are now emitted in the right place most of the time, but the change unfortunately introduced a regression, in that `BEGIN` blocks occurring just before the end of the enclosing block may appear below it instead. So this change may need to be reverted if it cannot be fixed before Perl 5.22. [perl #77452]

  B::Deparse no longer puts erroneous `local` here and there, such as for `LIST = tr/a//d`. [perl #119815]

  Adjacent `use` statements are no longer accidentally nested if one contains a `do` block. [perl

- *B::Op_private* has been upgraded from version 5.021005 to 5.021006.
  It now includes a hash named `%ops_using`, list all op types that use a particular private flag. [perl #115060]
- *CPAN::Meta* has been upgraded from version 2.142690 to 2.143240.
- *CPAN::Meta::Requirements* has been upgraded from version 2.128 to 2.130.
- *Devel::Peek* has been upgraded from version 1.18 to 1.19.
- *Digest::SHA* has been upgraded from version 5.92 to 5.93.
- *DynaLoader* has been upgraded from version 1.27 to 1.28.
- *Encode* has been upgraded from version 2.62 to 2.64.
- *experimental* has been upgraded from version 0.012 to 0.013.
- *Exporter* has been upgraded from version 5.71 to 5.72.
- *ExtUtils::MakeMaker* has been upgraded from version 6.98 to 7.02.
- *ExtUtils::Manifest* has been upgraded from version 1.68 to 1.69.
- *ExtUtils::ParseXS* has been upgraded from version 3.25 to 3.26.
- *HTTP::Tiny* has been upgraded from version 0.050 to 0.051.
- *I18N::Langinfo* has been upgraded from version 0.11 to 0.12.
- *IO::Socket* has been upgraded from version 1.37 to 1.38.
  Document the limitations of the connected() method. [perl #123096]
- *locale* has been upgraded from version 1.04 to 1.05.
- *Module::CoreList* has been upgraded from version 5.20141020 to 5.20141120.
- *overload* has been upgraded from version 1.23 to 1.24.
- *PerlIO::encoding* has been upgraded from version 0.19 to 0.20.
- *PerlIO::scalar* has been upgraded from version 0.19 to 0.20.
- *POSIX* has been upgraded from version 1.45 to 1.46.
- *re* has been upgraded from version 0.27 to 0.28.
- *Test::Harness* has been upgraded from version 3.33 to 3.34.
- *Test::Simple* has been upgraded from version 1.001008 to 1.301001_075.
- *Unicode::UCD* has been upgraded from version 0.58 to 0.59.
- *warnings* has been upgraded from version 1.28 to 1.29.
- *XSLoader* has been upgraded from version 0.18 to 0.19.

Documentation

Changes to Existing Documentation

perldata/Identifier parsing

- The syntax of single-character variable names has been brought up-to-date and more fully explained.

Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see *perldiag*.

New Diagnostics

New Warnings

- *Use of literal non-graphic characters in variable names is deprecated*
- A new `locale` warning category has been created, with the following warning messages currently in it:
  - *Locale '%s' may not work well.%s*
  - *Can't do %s("%s") on non-UTF-8 locale; resolved to "%s".*
- *Warning: unable to close filehandle %s properly: %s*
- The following two warnings for `tr///` used to be skipped if the transliteration contained wide characters, but now they occur regardless of whether there are wide characters or not:

  Useless use of /d modifier in transliteration operator

  Replacement list is longer than search list

Changes to Existing Diagnostics

- *Quantifier unexpected on zero-length expression in regex m/%s/*.

  This message has had the "<-- HERE" marker removed, as it was always placed at the end of the regular expression, regardless of where the problem actually occurred. [perl #122680]
- *Setting $/ to a reference to %s as a form of slurp is deprecated, treating as undef*

  This warning is now a default warning, like other deprecation warnings.

Configuration and Compilation

- *Configure* with `-Dmksymlinks` should now be faster. [perl #122002]
- As well as the gzip and bzip2 tarballs, this release has been made available as an xz utils compressed tarball.

Platform Support

Platform-Specific Notes

Win32

- In the experimental `:win32` layer, a crash in `open` was fixed. Also opening `/dev/null`, which works the Win32 Perl's normal `:unix` layer, was implemented for `:win32`. *[perl #122224]*
- A new makefile option, `USE_LONG_DOUBLE`, has been added to the Windows dmake makefile for gcc builds only. Set this to "define" if you want perl to use long doubles to give more accuracy and range for floating point numbers.

Internal Changes

- `screaminstr` has been removed. Although marked as public API, it is undocumented and has no usage in modern perl versions on CPAN Grep. Calling it has been fatal since 5.17.0.
- `newDEFSVOP`, `block_start`, `block_end` and `intro_my` have been added to the API.
- The internal `convert` function in *op.c* has been renamed `op_convert_list` and added to the API.
- `sv_magic` no longer forbids "ext" magic on read-only values. After all, perl can't know

whether the custom magic will modify the SV or not. [perl #123103]

- Starting in 5.21.6, accessing *"CvPADLIST" in perlapi* in an XSUB is forbidden. CvPADLIST has be reused for a different internal purpose for XSUBs. Guard all CvPADLIST expressions with `CvISXSUB()` if your code doesn't already block XSUB CV*s from going through optree CV* expecting code.

Selected Bug Fixes

- fchmod() and futimes() now set `$!` when they fail due to being passed a closed file handle. [perl #122703]
- Perl now comes with a corrected Unicode 7.0 for the erratum issued on October 21, 2014 (see *http://www.unicode.org/errata/#current_errata*), dealing with glyph shaping in Arabic.
- op_free() no longer crashes due to a stack overflow when freeing a deeply recursive op tree. [perl #108276]
- scalarvoid() would crash due to a stack overflow when processing a deeply recursive op tree. [perl #108276]
- In Perl 5.20.0, `$^N` accidentally had the internal UTF8 flag turned off if accessed from a code block within a regular expression, effectively UTF8-encoding the value. This has been fixed. [perl #123135]
- A failed `semctl` call no longer overwrites existing items on the stack, causing `(semctl(-1,0,0,0))[0]` to give an "uninitialized" warning.
- `else{foo()}` with no space before `foo` is now better at assigning the right line number to that statement. [perl #122695]
- Sometimes the assignment in `@array = split` gets optimised and `split` itself writes directly to the array. This caused a bug, preventing this assignment from being used in lvalue context. So `(@a=split//,"foo")=bar()` was an error. (This bug probably goes back to Perl 3, when the optimisation was added.) This optimisation, and the bug, started to happen in more cases in 5.21.5. It has now been fixed. [perl #123057]
- When argument lists that fail the checks installed by subroutine signatures, the resulting error messages now give the file and line number of the caller, not of the called subroutine. [perl #121374]
- Flip-flop operators (`..` and `...` in scalar context) used to maintain a separate state for each recursion level (the number of times the enclosing sub was called recursively), contrary to the documentation. Now each closure has one internal state for each flip-flop. [perl #122829]
- `use`, `no`, statement labels, special blocks (`BEGIN`) and pod are now permitted as the first thing in a `map` or `grep` block, the block after `print` or `say` (or other functions) returning a handle, and within `${...}`, `@{...}`, etc. [perl #122782]
- The repetition operator `x` now propagates lvalue context to its left-hand argument when used in contexts like `foreach`. That allows `for(($#that_array)x2) { ... }` to work as expected if the loop modifies `$_`.
- `(...) x ...` in scalar context used to corrupt the stack if one operand were an object with "x" overloading, causing erratic behaviour. [perl #121827]
- Assignment to a lexical scalar is often optimised away (as mentioned under *Performance Enhancements*). Various bugs related to this optimisation have been fixed. Certain operators on the right-hand side would sometimes fail to assign the value at all or assign the wrong value, or would call STORE twice or not at all on tied variables. The operators affected were `$foo++`, `$foo--`, and `-$foo` under `use integer`, `chomp`, `chr` and `setpgrp`.

- List assignments were sometimes buggy if the same scalar ended up on both sides of the assignment due to used of `tied`, `values` or `each`. The result would be the wrong value getting assigned.
- `setpgrp($nonzero)` (with one argument) was accidentally changed in 5.16 to mean `setpgrp(0)`. This has been fixed.
- `__SUB__` could return the wrong value or even corrupt memory under the debugger (the **-d** switch) and in subs containing `eval $string`.
- When `sub () { $var }` becomes inlinable, it now returns a different scalar each time, just as a non-inlinable sub would, though Perl still optimises the copy away in cases where it would make no observable difference.
- `my sub f () { $var }` and `sub () : attr { $var }` are no longer eligible for inlining. The former would crash; the latter would just throw the attributes away. An exception is made for the little-known ":method" attribute, which does nothing much.
- Inlining of subs with an empty prototype is now more consistent than before. Previously, a sub with multiple statements, all but the last optimised away, would be inlinable only if it were an anonymous sub containing a string `eval` or `state` declaration or closing over an outer lexical variable (or any anonymous sub under the debugger). Now any sub that gets folded to a single constant after statements have been optimised away is eligible for inlining. This applies to things like `sub () { jabber() if DEBUG; 42 }`.

  Some subroutines with an explicit `return` were being made inlinable, contrary to the documentation, Now `return` always prevents inlining.
- On some systems, such as VMS, `crypt` can return a non-ASCII string. If a scalar assigned to had contained a UTF8 string previously, then `crypt` would not turn off the UTF8 flag, thus corrupting the return value. This would happen with `$lexical = crypt ...`.
- `crypt` no longer calls `FETCH` twice on a tied first argument.
- An unterminated here-doc on the last line of a quote-like operator (`qq[${ <<END }]`, `/(?{ <<END })/`) no longer causes a double free. It started doing so in 5.18.
- Fixed two assertion failures introduced into `-DPERL_OP_PARENT` builds. [perl #108276]

Known Problems

- Builds on FreeBSD 10.x currently fail when compiling *POSIX*. A workaround is to specify `-Ui_fenv` when running `Configure`.

Errata From Previous Releases

- Due to a mistake in the string-copying logic, copying the value of a state variable could instead steal the value and undefine the variable. This bug, introduced in 5.20, would happen mostly for long strings (1250 chars or more), but could happen for any strings under builds with copy-on-write disabled. [perl #123029]

  This bug was actually fixed in 5.21.5, but it was not until after that release that this bug, and the fact that it had been fixed, were discovered.
- If a named sub tries to access a scalar declared in an outer anonymous sub, the variable is not available, so the named sub gets its own undefined scalar. In 5.10, attempts to take a reference to the variable (`\$that_variable`) began returning a reference to a *copy* of it instead. This was accidentally fixed in 5.21.4, but the bug and its fix were not noticed till now.

Acknowledgements

Perl 5.21.6 represents approximately 4 weeks of development since Perl 5.21.5 and contains approximately 60,000 lines of changes across 920 files from 25 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 48,000 lines of changes to 630 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.21.6:

Aaron Crane, Abigail, Andrew Fresh, Andy Dougherty, Brian Fraser, Chad Granum, Chris 'BinGOs' Williams, Craig A. Berry, Daniel Dragan, David Mitchell, Doug Bell, Father Chrysostomos, Glenn D. Golden, James E Keenan, Jarkko Hietaniemi, Jim Cromie, Karen Etheridge, Karl Williamson, Lukas Mai, Ricardo Signes, Shlomi Fish, Slaven Rezic, Steve Hay, Tony Cook, Yaroslav Kuzmin.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

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 https://rt.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.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

SEE ALSO

The *Changes* file for an explanation of how to view 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.