NAME

perldelta - what is new for perl v5.26.0

DESCRIPTION

This document describes the differences between the 5.24.0 release and the 5.26.0 release.

Notice

This release includes three updates with widespread effects:

* "." no longer in @INC

  For security reasons, the current directory (".") is no longer included by default at the end of the module search path (@INC). This may have widespread implications for the building, testing and installing of modules, and for the execution of scripts. See the section *Removal of the current directory (".") from @INC* for the full details.

* do may now warn

  do now gives a deprecation warning when it fails to load a file which it would have loaded had "." been in @INC.

* In regular expression patterns, a literal left brace "{" should be escaped

  See *Unescaped literal "{" characters in regular expression patterns are no longer permissible* .

Core Enhancements

Lexical subroutines are no longer experimental

Using the `lexical_subs` feature introduced in v5.18 no longer emits a warning. Existing code that disables the `experimental::lexical_subs` warning category that the feature previously used will continue to work. The `lexical_subs` feature has no effect; all Perl code can use lexical subroutines, regardless of what feature declarations are in scope.

Indented Here-documents

This adds a new modifier "~" to here-docs that tells the parser that it should look for `/^\s*$DELIM\n/` as the closing delimiter.

These syntaxes are all supported:

```
<<~EOF;
<<~\EOF;
<<~'EOF';
<<~"EOF";
<<~`EOF`;
<<~ 'EOF';
<<~ "EOF";
<<~ `EOF`;
```

The "~" modifier will strip, from each line in the here-doc, the same whitespace that appears before the delimiter.

Newlines will be copied as-is, and lines that don't include the proper beginning whitespace will cause perl to croak.

For example:

```
if (1) {
  print <<~EOF;
    Hello there
    EOF
```

```
        }
```

prints "Hello there\n" with no leading whitespace.

New regular expression modifier /xx

Specifying two `"x"` characters to modify a regular expression pattern does everything that a single one does, but additionally TAB and SPACE characters within a bracketed character class are generally ignored and can be added to improve readability, like `/[ ^ A-Z d-f p-x ]/xx`. Details are at *"/x and /xx" in perlre*.

@{^CAPTURE}, %{^CAPTURE}, and %{^CAPTURE_ALL}

`@{^CAPTURE}` exposes the capture buffers of the last match as an array. So `$1` is `${^CAPTURE}[0]` . This is a more efficient equivalent to code like `substr($matched_string,$-[0],$+[0]-$-[0])`, and you don't have to keep track of the `$matched_string` either. This variable has no single character equivalent. Note that, like the other regex magic variables, the contents of this variable is dynamic; if you wish to store it beyond the lifetime of the match you must copy it to another array.

`%{^CAPTURE}` is equivalent to `%+` (*i.e.*, named captures). Other than being more self-documenting there is no difference between the two forms.

`%{^CAPTURE_ALL}` is equivalent to `%-` (*i.e.*, all named captures). Other than being more self-documenting there is no difference between the two forms.

Declaring a reference to a variable

As an experimental feature, Perl now allows the referencing operator to come after *`my()`*, *`state()`*, *`our()`*, or *`local()`*. This syntax must be enabled with `use feature 'declared_refs'`. It is experimental, and will warn by default unless `no warnings 'experimental::refaliasing'` is in effect. It is intended mainly for use in assignments to references. For example:

```
    use experimental 'refaliasing', 'declared_refs';
    my \$a = \$b;
```

See *"Assigning to References" in perlref* for more details.

Unicode 9.0 is now supported

A list of changes is at *http://www.unicode.org/versions/Unicode9.0.0/*. Modules that are shipped with core Perl but not maintained by p5p do not necessarily support Unicode 9.0. *Unicode::Normalize* does work on 9.0.

Use of \p{script} uses the improved Script_Extensions property

Unicode 6.0 introduced an improved form of the Script (`sc`) property, and called it Script_Extensions (`scx`). Perl now uses this improved version when a property is specified as just `\p{`*`script`*`}`. This should make programs more accurate when determining if a character is used in a given script, but there is a slight chance of breakage for programs that very specifically needed the old behavior. The meaning of compound forms, like `\p{sc=`*`script`*`}` are unchanged. See *"Scripts" in perlunicode*.

Perl can now do default collation in UTF-8 locales on platforms that support it

Some platforms natively do a reasonable job of collating and sorting in UTF-8 locales. Perl now works with those. For portability and full control, *Unicode::Collate* is still recommended, but now you may not need to do anything special to get good-enough results, depending on your application. See *"Category `LC_COLLATE`: Collation: Text Comparisons and Sorting" in perllocale*.

Better locale collation of strings containing embedded NUL characters

In locales that have multi-level character weights, `NUL`s are now ignored at the higher priority ones. There are still some gotchas in some strings, though. See *"Collation of strings containing embedded*

`NUL` characters" in perllocale.

CORE subroutines for hash and array functions callable via reference

The hash and array functions in the `CORE` namespace (`keys`, `each`, `values`, `push`, `pop`, `shift`, `unshift` and `splice`) can now be called with ampersand syntax (`&CORE::keys(\%hash`) and via reference (`my $k = \&CORE::keys; $k->(\%hash)`). Previously they could only be used when inlined.

New Hash Function For 64-bit Builds

We have switched to a hybrid hash function to better balance performance for short and long keys.

For short keys, 16 bytes and under, we use an optimised variant of One At A Time Hard, and for longer keys we use Siphash 1-3. For very long keys this is a big improvement in performance. For shorter keys there is a modest improvement.

Security

Removal of the current directory (".") from @INC

The perl binary includes a default set of paths in `@INC`. Historically it has also included the current directory (`"."`) as the final entry, unless run with taint mode enabled (`perl -T`). While convenient, this has security implications: for example, where a script attempts to load an optional module when its current directory is untrusted (such as */tmp*), it could load and execute code from under that directory.

Starting with v5.26, `"."` is always removed by default, not just under tainting. This has major implications for installing modules and executing scripts.

The following new features have been added to help ameliorate these issues.

* *Configure -Udefault_inc_excludes_dot*

There is a new *Configure* option, `default_inc_excludes_dot` (enabled by default) which builds a perl executable without `"."`; unsetting this option using `-U` reverts perl to the old behaviour. This may fix your path issues but will reintroduce all the security concerns, so don't build a perl executable like this unless you're *really* confident that such issues are not a concern in your environment.

* `PERL_USE_UNSAFE_INC`

There is a new environment variable recognised by the perl interpreter. If this variable has the value 1 when the perl interpreter starts up, then `"."` will be automatically appended to `@INC` (except under tainting).

This allows you restore the old perl interpreter behaviour on a case-by-case basis. But note that this is intended to be a temporary crutch, and this feature will likely be removed in some future perl version. It is currently set by the `cpan` utility and `Test::Harness` to ease installation of CPAN modules which have not been updated to handle the lack of dot. Once again, don't use this unless you are sure that this will not reintroduce any security concerns.

* A new deprecation warning issued by `do`.

While it is well-known that `use` and `require` use `@INC` to search for the file to load, many people don't realise that `do "file"` also searches `@INC` if the file is a relative path. With the removal of `"."`, a simple `do "file.pl"` will fail to read in and execute `file.pl` from the current directory. Since this is commonly expected behaviour, a new deprecation warning is now issued whenever `do` fails to load a file which it otherwise would have found if a dot had been in `@INC`.

Here are some things script and module authors may need to do to make their software work in the new regime.

* Script authors

If the issue is within your own code (rather than within included modules), then you have two main options. Firstly, if you are confident that your script will only be run within a trusted directory (under which you expect to find trusted files and modules), then add `"."` back into the path; *e.g.*:

```
BEGIN {
    my $dir = "/some/trusted/directory";
    chdir $dir or die "Can't chdir to $dir: $!\n";
    # safe now
    push @INC, '.';
}

use "Foo::Bar"; # may load /some/trusted/directory/Foo/Bar.pm
do "config.pl"; # may load /some/trusted/directory/config.pl
```

On the other hand, if your script is intended to be run from within untrusted directories (such as */tmp*), then your script suddenly failing to load files may be indicative of a security issue. You most likely want to replace any relative paths with full paths; for example,

```
do "foo_config.pl"
```

might become

```
do "$ENV{HOME}/foo_config.pl"
```

If you are absolutely certain that you want your script to load and execute a file from the current directory, then use a `./` prefix; for example:

```
do "./foo_config.pl"
```

* Installing and using CPAN modules

If you install a CPAN module using an automatic tool like `cpan`, then this tool will itself set the `PERL_USE_UNSAFE_INC` environment variable while building and testing the module, which may be sufficient to install a distribution which hasn't been updated to be dot-aware. If you want to install such a module manually, then you'll need to replace the traditional invocation:

```
perl Makefile.PL && make && make test && make install
```

with something like

```
(export PERL_USE_UNSAFE_INC=1; \
 perl Makefile.PL && make && make test && make install)
```

Note that this only helps build and install an unfixed module. It's possible for the tests to pass (since they were run under `PERL_USE_UNSAFE_INC=1`), but for the module itself to fail to perform correctly in production. In this case, you may have to temporarily modify your script until a fixed version of the module is released. For example:

```
use Foo::Bar;
{
    local @INC = (@INC, '.');
    # assuming read_config() needs '.' in @INC
    $config = Foo::Bar->read_config();
}
```

This is only rarely expected to be necessary. Again, if doing this, assess the resultant risks first.

* Module Authors

If you maintain a CPAN distribution, it may need updating to run in a dotless environment.

Although `cpan` and other such tools will currently set the `PERL_USE_UNSAFE_INC` during module build, this is a temporary workaround for the set of modules which rely on `"."` being in `@INC` for installation and testing, and this may mask deeper issues. It could result in a module which passes tests and installs, but which fails at run time.

During build, test, and install, it will normally be the case that any perl processes will be executing directly within the root directory of the untarred distribution, or a known subdirectory of that, such as *t/*. It may well be that *Makefile.PL* or *t/foo.t* will attempt to include local modules and configuration files using their direct relative filenames, which will now fail.

However, as described above, automatic tools like *cpan* will (for now) set the `PERL_USE_UNSAFE_INC` environment variable, which introduces dot during a build.

This makes it likely that your existing build and test code will work, but this may mask issues with your code which only manifest when used after install. It is prudent to try and run your build process with that variable explicitly disabled:

```
    (export PERL_USE_UNSAFE_INC=0; \
     perl Makefile.PL && make && make test && make install)
```

This is more likely to show up any potential problems with your module's build process, or even with the module itself. Fixing such issues will ensure both that your module can again be installed manually, and that it will still build once the `PERL_USE_UNSAFE_INC` crutch goes away.

When fixing issues in tests due to the removal of dot from `@INC`, reinsertion of dot into `@INC` should be performed with caution, for this too may suppress real errors in your runtime code. You are encouraged wherever possible to apply the aforementioned approaches with explicit absolute/relative paths, or to relocate your needed files into a subdirectory and insert that subdirectory into `@INC` instead.

If your runtime code has problems under the dotless `@INC`, then the comments above on how to fix for script authors will mostly apply here too. Bear in mind though that it is considered bad form for a module to globally add a dot to `@INC`, since it introduces both a security risk and hides issues of accidentally requiring dot in `@INC`, as explained above.

Escaped colons and relative paths in PATH

On Unix systems, Perl treats any relative paths in the `PATH` environment variable as tainted when starting a new process. Previously, it was allowing a backslash to escape a colon (unlike the OS), consequently allowing relative paths to be considered safe if the PATH was set to something like `/\:.`. The check has been fixed to treat `"."` as tainted in that example.

New -Di switch is now required for PerlIO debugging output

This is used for debugging of code within PerlIO to avoid recursive calls. Previously this output would be sent to the file specified by the `PERLIO_DEBUG` environment variable if perl wasn't running setuid and the `-T` or `-t` switches hadn't been parsed yet.

If perl performed output at a point where it hadn't yet parsed its switches this could result in perl creating or overwriting the file named by `PERLIO_DEBUG` even when the `-T` switch had been supplied.

Perl now requires the `-Di` switch to be present before it will produce PerlIO debugging output. By default this is written to `stderr`, but can optionally be redirected to a file by setting the `PERLIO_DEBUG` environment variable.

If perl is running setuid or the `-T` switch was supplied, `PERLIO_DEBUG` is ignored and the debugging output is sent to `stderr` as for any other `-D` switch.

Incompatible Changes

Unescaped literal "{" characters in regular expression patterns are no longer permissible

You have to now say something like `"\{"` or `"[{]"` to specify to match a LEFT CURLY BRACKET; otherwise, it is a fatal pattern compilation error. This change will allow future extensions to the language.

These have been deprecated since v5.16, with a deprecation message raised for some uses starting in v5.22. Unfortunately, the code added to raise the message was buggy and failed to warn in some cases where it should have. Therefore, enforcement of this ban for these cases is deferred until Perl 5.30, but the code has been fixed to raise a default-on deprecation message for them in the meantime.

Some uses of literal `"{"` occur in contexts where we do not foresee the meaning ever being anything but the literal, such as the very first character in the pattern, or after a `"|"` meaning alternation. Thus

```
qr/{fee|{fie/
```

matches either of the strings `{fee` or `{fie`. To avoid forcing unnecessary code changes, these uses do not need to be escaped, and no warning is raised about them, and there are no current plans to change this.

But it is always correct to escape `"{"`, and the simple rule to remember is to always do so.

See *Unescaped left brace in regex is illegal here*.

scalar(%hash) return signature changed

The value returned for `scalar(%hash)` will no longer show information about the buckets allocated in the hash. It will simply return the count of used keys. It is thus equivalent to `0+keys(%hash)`.

A form of backward compatibility is provided via *Hash::Util::bucket_ratio()* which provides the same behavior as `scalar(%hash)` provided in Perl 5.24 and earlier.

keys returned from an lvalue subroutine

`keys` returned from an lvalue subroutine can no longer be assigned to in list context.

```
    sub foo : lvalue { keys(%INC) }
    (foo) = 3; # death
    sub bar : lvalue { keys(@_) }
    (bar) = 3; # also an error
```

This makes the lvalue sub case consistent with `(keys %hash) = ...` and `(keys @_) = ...`, which are also errors. *[perl #128187]*

The ${^ENCODING} facility has been removed

The special behaviour associated with assigning a value to this variable has been removed. As a consequence, the *encoding* pragma's default mode is no longer supported. If you still need to write your source code in encodings other than UTF-8, use a source filter such as *Filter::Encoding* on CPAN or *encoding*'s `Filter` option.

POSIX::tmpnam() has been removed

The fundamentally unsafe `tmpnam()` interface was deprecated in Perl 5.22 and has now been removed. In its place, you can use, for example, the *File::Temp* interfaces.

require ::Foo::Bar is now illegal.

Formerly, `require ::Foo::Bar` would try to read */Foo/Bar.pm*. Now any bareword require which starts with a double colon dies instead.

Literal control character variable names are no longer permissible

A variable name may no longer contain a literal control character under any circumstances. These previously were allowed in single-character names on ASCII platforms, but have been deprecated there since Perl 5.20. This affects things like `$\cT`, where \cT is a literal control (such as a `NAK` or `NEGATIVE ACKNOWLEDGE` character) in the source code.

NBSP is no longer permissible in \N{...}

The name of a character may no longer contain non-breaking spaces. It has been deprecated to do so since Perl 5.22.

Deprecations

String delimiters that aren't stand-alone graphemes are now deprecated

For Perl to eventually allow string delimiters to be Unicode grapheme clusters (which look like a single character, but may be a sequence of several ones), we have to stop allowing a single character delimiter that isn't a grapheme by itself. These are unlikely to exist in actual code, as they would typically display as attached to the character in front of them.

\cX that maps to a printable is no longer deprecated

This means we have no plans to remove this feature. It still raises a warning, but only if syntax warnings are enabled. The feature was originally intended to be a way to express non-printable characters that don't have a mnemonic (`\t` and `\n` are mnemonics for two non-printable characters, but most non-printables don't have a mnemonic.) But the feature can be used to specify a few printable characters, though those are more clearly expressed as the printable itself. See *http://www.nntp.perl.org/group/perl.perl5.porters/2017/02/msg242944.html*.

Performance Enhancements

- A hash in boolean context is now sometimes faster, *e.g.*

  ```
  if (!%h) { ... }
  ```

  This was already special-cased, but some cases were missed (such as `grep %$_, @AoH`), and even the ones which weren't have been improved.

* New Faster Hash Function on 64 bit builds

  We use a different hash function for short and long keys. This should improve performance and security, especially for long keys.

* readline is faster

  Reading from a file line-by-line with `readline()` or <> should now typically be faster due to a better implementation of the code that searches for the next newline character.

- Assigning one reference to another, *e.g.* `$ref1 = $ref2` has been optimized in some cases.

- Remove some exceptions to creating Copy-on-Write strings. The string buffer growth algorithm has been slightly altered so that you're less likely to encounter a string which can't be COWed.

- Better optimise array and hash assignment: where an array or hash appears in the LHS of a list assignment, such as `(..., @a) = (...);`, it's likely to be considerably faster, especially if it involves emptying the array/hash. For example, this code runs about a third faster compared to Perl 5.24.0:

  ```
  my @a;
  for my $i (1..10_000_000) {
      @a = (1,2,3);
      @a = ();
  }
  ```

- Converting a single-digit string to a number is now substantially faster.
- The `split` builtin is now slightly faster in many cases: in particular for the two specially-handled forms

  ```
  my    @a = split ...;
  local @a = split ...;
  ```

- The rather slow implementation for the experimental subroutine signatures feature has been made much faster; it is now comparable in speed with the traditional `my ($a, $b, @c) = @_`.
- Bareword constant strings are now permitted to take part in constant folding. They were originally exempted from constant folding in August 1999, during the development of Perl 5.6, to ensure that `use strict "subs"` would still apply to bareword constants. That has now been accomplished a different way, so barewords, like other constants, now gain the performance benefits of constant folding.

  This also means that void-context warnings on constant expressions of barewords now report the folded constant operand, rather than the operation; this matches the behaviour for non-bareword constants.

Modules and Pragmata

Updated Modules and Pragmata

- IO::Compress has been upgraded from version 2.069 to 2.074.
- *Archive::Tar* has been upgraded from version 2.04 to 2.24.
- *arybase* has been upgraded from version 0.11 to 0.12.
- *attributes* has been upgraded from version 0.27 to 0.29.

  The deprecation message for the `:unique` and `:locked` attributes now mention that they will disappear in Perl 5.28.
- *B* has been upgraded from version 1.62 to 1.68.
- *B::Concise* has been upgraded from version 0.996 to 0.999.

  Its output is now more descriptive for `op_private` flags.
- *B::Debug* has been upgraded from version 1.23 to 1.24.
- *B::Deparse* has been upgraded from version 1.37 to 1.40.
- *B::Xref* has been upgraded from version 1.05 to 1.06.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *base* has been upgraded from version 2.23 to 2.25.
- *bignum* has been upgraded from version 0.42 to 0.47.
- *Carp* has been upgraded from version 1.40 to 1.42.
- *charnames* has been upgraded from version 1.43 to 1.44.
- *Compress::Raw::Bzip2* has been upgraded from version 2.069 to 2.074.
- *Compress::Raw::Zlib* has been upgraded from version 2.069 to 2.074.
- *Config::Perl::V* has been upgraded from version 0.25 to 0.28.
- *CPAN* has been upgraded from version 2.11 to 2.18.

- *CPAN::Meta* has been upgraded from version 2.150005 to 2.150010.
- *Data::Dumper* has been upgraded from version 2.160 to 2.167.

  The XS implementation now supports Deparse.
- *DB_File* has been upgraded from version 1.835 to 1.840.
- *Devel::Peek* has been upgraded from version 1.23 to 1.26.
- *Devel::PPPort* has been upgraded from version 3.32 to 3.35.
- *Devel::SelfStubber* has been upgraded from version 1.05 to 1.06.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *diagnostics* has been upgraded from version 1.34 to 1.36.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *Digest* has been upgraded from version 1.17 to 1.17_01.
- *Digest::MD5* has been upgraded from version 2.54 to 2.55.
- *Digest::SHA* has been upgraded from version 5.95 to 5.96.
- *DynaLoader* has been upgraded from version 1.38 to 1.42.
- *Encode* has been upgraded from version 2.80 to 2.88.
- *encoding* has been upgraded from version 2.17 to 2.19.

  This module's default mode is no longer supported. It now dies when imported, unless the `Filter` option is being used.
- *encoding::warnings* has been upgraded from version 0.12 to 0.13.

  This module is no longer supported. It emits a warning to that effect and then does nothing.
- *Errno* has been upgraded from version 1.25 to 1.28.

  It now documents that using `%!` automatically loads Errno for you.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *ExtUtils::Embed* has been upgraded from version 1.33 to 1.34.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *ExtUtils::MakeMaker* has been upgraded from version 7.10_01 to 7.24.
- *ExtUtils::Miniperl* has been upgraded from version 1.05 to 1.06.
- *ExtUtils::ParseXS* has been upgraded from version 3.31 to 3.34.
- *ExtUtils::Typemaps* has been upgraded from version 3.31 to 3.34.
- *feature* has been upgraded from version 1.42 to 1.47.
- *File::Copy* has been upgraded from version 2.31 to 2.32.
- *File::Fetch* has been upgraded from version 0.48 to 0.52.
- *File::Glob* has been upgraded from version 1.26 to 1.28.

  It now Issues a deprecation message for `File::Glob::glob()`.
- *File::Spec* has been upgraded from version 3.63 to 3.67.
- *FileHandle* has been upgraded from version 2.02 to 2.03.

- *Filter::Simple* has been upgraded from version 0.92 to 0.93.

  It no longer treats `no MyFilter` immediately following `use MyFilter` as end-of-file. *[perl #107726]*

- *Getopt::Long* has been upgraded from version 2.48 to 2.49.
- *Getopt::Std* has been upgraded from version 1.11 to 1.12.
- *Hash::Util* has been upgraded from version 0.19 to 0.22.
- *HTTP::Tiny* has been upgraded from version 0.056 to 0.070.

  Internal 599-series errors now include the redirect history.

- *I18N::LangTags* has been upgraded from version 0.40 to 0.42.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*

- *IO* has been upgraded from version 1.36 to 1.38.
- *IO::Socket::IP* has been upgraded from version 0.37 to 0.38.
- *IPC::Cmd* has been upgraded from version 0.92 to 0.96.
- *IPC::SysV* has been upgraded from version 2.06_01 to 2.07.
- *JSON::PP* has been upgraded from version 2.27300 to 2.27400_02.
- *lib* has been upgraded from version 0.63 to 0.64.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*

- *List::Util* has been upgraded from version 1.42_02 to 1.46_02.
- *Locale::Codes* has been upgraded from version 3.37 to 3.42.
- *Locale::Maketext* has been upgraded from version 1.26 to 1.28.
- *Locale::Maketext::Simple* has been upgraded from version 0.21 to 0.21_01.
- *Math::BigInt* has been upgraded from version 1.999715 to 1.999806.
- *Math::BigInt::FastCalc* has been upgraded from version 0.40 to 0.5005.
- *Math::BigRat* has been upgraded from version 0.260802 to 0.2611.
- *Math::Complex* has been upgraded from version 1.59 to 1.5901.
- *Memoize* has been upgraded from version 1.03 to 1.03_01.
- *Module::CoreList* has been upgraded from version 5.20170420 to 5.20170530.
- *Module::Load::Conditional* has been upgraded from version 0.64 to 0.68.
- *Module::Metadata* has been upgraded from version 1.000031 to 1.000033.
- *mro* has been upgraded from version 1.18 to 1.20.
- *Net::Ping* has been upgraded from version 2.43 to 2.55.

  IPv6 addresses and `AF_INET6` sockets are now supported, along with several other enhancements.

- *NEXT* has been upgraded from version 0.65 to 0.67.
- *Opcode* has been upgraded from version 1.34 to 1.39.
- *open* has been upgraded from version 1.10 to 1.11.

- *OS2::Process* has been upgraded from version 1.11 to 1.12.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*

- *overload* has been upgraded from version 1.26 to 1.28.

  Its compilation speed has been improved slightly.

- *parent* has been upgraded from version 0.234 to 0.236.

- *perl5db.pl* has been upgraded from version 1.50 to 1.51.

  It now ignores */dev/tty* on non-Unix systems. *[perl #113960]*

- *Perl::OSType* has been upgraded from version 1.009 to 1.010.

- *perlfaq* has been upgraded from version 5.021010 to 5.021011.

- *PerlIO* has been upgraded from version 1.09 to 1.10.

- *PerlIO::encoding* has been upgraded from version 0.24 to 0.25.

- *PerlIO::scalar* has been upgraded from version 0.24 to 0.26.

- *Pod::Checker* has been upgraded from version 1.60 to 1.73.

- *Pod::Functions* has been upgraded from version 1.10 to 1.11.

- *Pod::Html* has been upgraded from version 1.22 to 1.2202.

- *Pod::Perldoc* has been upgraded from version 3.25_02 to 3.28.

- *Pod::Simple* has been upgraded from version 3.32 to 3.35.

- *Pod::Usage* has been upgraded from version 1.68 to 1.69.

- *POSIX* has been upgraded from version 1.65 to 1.76.

  This remedies several defects in making its symbols exportable. *[perl #127821]*

  The `POSIX::tmpnam()` interface has been removed, see *POSIX::tmpnam() has been removed*.

  The following deprecated functions have been removed:

  ```
  POSIX::isalnum
  POSIX::isalpha
  POSIX::iscntrl
  POSIX::isdigit
  POSIX::isgraph
  POSIX::islower
  POSIX::isprint
  POSIX::ispunct
  POSIX::isspace
  POSIX::isupper
  POSIX::isxdigit
  POSIX::tolower
  POSIX::toupper
  ```

  Trying to import POSIX subs that have no real implementations (like `POSIX::atend()`) now fails at import time, instead of waiting until runtime.

- *re* has been upgraded from version 0.32 to 0.34

  This adds support for the new */xx* regular expression pattern modifier, and a change to the *`use re 'strict'`* experimental feature. When `re 'strict'` is enabled, a warning now will be generated for all unescaped uses of the two characters `"}"` and `"]"` in regular expression patterns (outside bracketed character classes) that are taken literally. This brings them more in line with the `")"` character which is always a metacharacter unless escaped. Being a metacharacter only sometimes, depending on an action at a distance, can lead to silently having the pattern mean something quite different than was intended, which the `re 'strict'` mode is intended to minimize.

- *Safe* has been upgraded from version 2.39 to 2.40.
- *Scalar::Util* has been upgraded from version 1.42_02 to 1.46_02.
- *Storable* has been upgraded from version 2.56 to 2.62.

  Fixes *[perl #130098]*.
- *Symbol* has been upgraded from version 1.07 to 1.08.
- *Sys::Syslog* has been upgraded from version 0.33 to 0.35.
- *Term::ANSIColor* has been upgraded from version 4.04 to 4.06.
- *Term::ReadLine* has been upgraded from version 1.15 to 1.16.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *Test* has been upgraded from version 1.28 to 1.30.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *Test::Harness* has been upgraded from version 3.36 to 3.38.
- *Test::Simple* has been upgraded from version 1.001014 to 1.302073.
- *Thread::Queue* has been upgraded from version 3.09 to 3.12.
- *Thread::Semaphore* has been upgraded from 2.12 to 2.13.

  Added the `down_timed` method.
- *threads* has been upgraded from version 2.07 to 2.15.
- *threads::shared* has been upgraded from version 1.51 to 1.56.
- *Tie::Hash::NamedCapture* has been upgraded from version 0.09 to 0.10.
- *Time::HiRes* has been upgraded from version 1.9733 to 1.9741.

  It now builds on systems with C++11 compilers (such as G++ 6 and Clang++ 3.9).

  Now uses `clockid_t`.
- *Time::Local* has been upgraded from version 1.2300 to 1.25.
- *Unicode::Collate* has been upgraded from version 1.14 to 1.19.
- *Unicode::UCD* has been upgraded from version 0.64 to 0.68.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *version* has been upgraded from version 0.9916 to 0.9917.
- *VMS::DCLsym* has been upgraded from version 1.06 to 1.08.

  It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*
- *warnings* has been upgraded from version 1.36 to 1.37.
- *XS::Typemap* has been upgraded from version 0.14 to 0.15.
- *XSLoader* has been upgraded from version 0.21 to 0.27.

Fixed a security hole in which binary files could be loaded from a path outside of `@INC`.

It now uses 3-arg `open()` instead of 2-arg `open()`. *[perl #130122]*

Documentation

New Documentation

perldeprecation

This file documents all upcoming deprecations, and some of the deprecations which already have been removed. The purpose of this documentation is two-fold: document what will disappear, and by which version, and serve as a guide for people dealing with code which has features that no longer work after an upgrade of their perl.

Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to *perlbug@perl.org*.

Additionally, all references to Usenet have been removed, and the following selected changes have been made:

perlfunc

- Removed obsolete text about *`defined()`* on aggregates that should have been deleted earlier, when the feature was removed.
- Corrected documentation of *`eval()`*, and *`evalbytes()`*.
- Clarified documentation of *`seek()`*, *`tell()`* and *`sysseek()`* emphasizing that positions are in bytes and not characters. *[perl #128607]*
- Clarified documentation of *`sort()`* concerning the variables $a and $b.
- In *`split()`* noted that certain pattern modifiers are legal, and added a caution about its use in Perls before v5.11.
- Removed obsolete documentation of *`study()`*, noting that it is now a no-op.
- Noted that *`vec()`* doesn't work well when the string contains characters whose code points are above 255.

perlguts

- Added advice on *formatted printing of operands of `Size_t` and `SSize_t`*

perlhack

- Clarify what editor tab stop rules to use, and note that we are migrating away from using tabs, replacing them with sequences of SPACE characters.

perlhacktips

- Give another reason to use `cBOOL` to cast an expression to boolean.
- Note that the macros TRUE and FALSE are available to express boolean values.

perlinterp

- *perlinterp* has been expanded to give a more detailed example of how to hunt around in the parser for how a given operator is handled.

perllocale

- Some locales aren't compatible with Perl. Note that these can cause core dumps.

perlmod

- Various clarifications have been added.

perlmodlib

- Updated the site mirror list.

perlobj

- Added a section on calling methods using their fully qualified names.
- Do not discourage manual `@ISA`.

perlootut

- Mention `Moo` more.

perlop

- Note that white space must be used for quoting operators if the delimiter is a word character (*i.e.*, matches `\w`).
- Clarify that in regular expression patterns delimited by single quotes, no variable interpolation is done.

perlre

- The first part was extensively rewritten to incorporate various basic points, that in earlier versions were mentioned in sort of an appendix on Version 8 regular expressions.
- Note that it is common to have the `/x` modifier and forget that this means that `"#"` has to be escaped.

perlretut

- Add introductory material.
- Note that a metacharacter occurring in a context where it can't mean that, silently loses its meta-ness and matches literally. *`use re 'strict'`* can catch some of these.

perlunicode

- Corrected the text about Unicode BYTE ORDER MARK handling.
- Updated the text to correspond with changes in Unicode UTS#18, concerning regular expressions, and Perl compatibility with what it says.

perlvar

- Document `@ISA`. It was documented in other places, but not in *perlvar*.

Diagnostics

New Diagnostics

New Errors

- *A signature parameter must start with `'$'`, `'@'` or `'%'`*
- *Bareword in require contains "%s"*
- *Bareword in require maps to empty filename*
- *Bareword in require maps to disallowed filename "%s"*
- *Bareword in require must not start with a double-colon: "%s"*
- *%s: command not found*

  (A) You've accidentally run your script through **bash** or another shell instead of Perl. Check the `#!` line, or manually feed your script into Perl yourself. The `#!` line at the top of your file could look like:

```
#!/usr/bin/perl
```

- *%s: command not found: %s*

  (A) You've accidentally run your script through **zsh** or another shell instead of Perl. Check the `#!` line, or manually feed your script into Perl yourself. The `#!` line at the top of your file could look like:

  ```
  #!/usr/bin/perl
  ```

- *The experimental declared_refs feature is not enabled*

  (F) To declare references to variables, as in `my \%x`, you must first enable the feature:

  ```
      no warnings "experimental::declared_refs";
      use feature "declared_refs";
  ```

  See *Declaring a reference to a variable*.

- *Illegal character following sigil in a subroutine signature*
- *Indentation on line %d of here-doc doesn't match delimiter*
- *Infinite recursion via empty pattern*.

  Using the empty pattern (which re-executes the last successfully-matched pattern) inside a code block in another regex, as in `/(?{ s!!new! })/`, has always previously yielded a segfault. It now produces this error.

- *Malformed UTF-8 string in "%s"*
- *Multiple slurpy parameters not allowed*
- *`'#'` not allowed immediately following a sigil in a subroutine signature*
- *panic: unknown OA_*: %x*
- *Unescaped left brace in regex is illegal here*

  Unescaped left braces are now illegal in some contexts in regular expression patterns. In other contexts, they are still just deprecated; they will be illegal in Perl 5.30.

- *Version control conflict marker*

  (F) The parser found a line starting with <<<<<<<, >>>>>>>, or =======. These may be left by a version control system to mark conflicts after a failed merge operation.

New Warnings

- *Can't determine class of operator %s, assuming `BASEOP`*
- *Declaring references is experimental*

  (S experimental::declared_refs) This warning is emitted if you use a reference constructor on the right-hand side of `my()`, `state()`, `our()`, or `local()`. Simply suppress the warning if you want to use the feature, but know that in doing so you are taking the risk of using an experimental feature which may change or be removed in a future Perl version:

  ```
      no warnings "experimental::declared_refs";
      use feature "declared_refs";
      $fooref = my \$foo;
  ```

  See *Declaring a reference to a variable*.

- *do "%s" failed, '.' is no longer in @INC*

  Since `"."` is now removed from `@INC` by default, do will now trigger a warning recommending to fix the do statement.

- *`File::Glob::glob()` will disappear in perl 5.30. Use `File::Glob::bsd_glob()` instead.*

- *Unescaped literal '%c' in regex; marked by <-- HERE in m/%s/*

- *Use of unassigned code point or non-standalone grapheme for a delimiter will be a fatal error starting in Perl 5.30*

  See *Deprecations*

Changes to Existing Diagnostics

- When a `require` fails, we now do not provide `@INC` when the `require` is for a file instead of a module.

- When `@INC` is not scanned for a `require` call, we no longer display `@INC` to avoid confusion.

- *Attribute "locked" is deprecated, and will disappear in Perl 5.28*

  This existing warning has had the *and will disappear* text added in this release.

- *Attribute "unique" is deprecated, and will disappear in Perl 5.28*

  This existing warning has had the *and will disappear* text added in this release.

- Calling POSIX::%s() is deprecated

  This warning has been removed, as the deprecated functions have been removed from POSIX.

- *Constants from lexical variables potentially modified elsewhere are deprecated. This will not be allowed in Perl 5.32*

  This existing warning has had the *this will not be allowed* text added in this release.

- *Deprecated use of `my()` in false conditional. This will be a fatal error in Perl 5.30*

  This existing warning has had the *this will be a fatal error* text added in this release.

- *`dump()` better written as `CORE::dump()`. `dump()` will no longer be available in Perl 5.30*

  This existing warning has had the *no longer be available* text added in this release.

- *Experimental %s on scalar is now forbidden*

  This message is now followed by more helpful text. *[perl #127976]*

- Experimental "%s" subs not enabled

  This warning was been removed, as lexical subs are no longer experimental.

- Having more than one /%c regexp modifier is deprecated

  This deprecation warning has been removed, since `/xx` now has a new meaning.

- *%s() is deprecated on `:utf8` handles. This will be a fatal error in Perl 5.30* .

  where "%s" is one of `sysread`, `recv`, `syswrite`, or `send`.

  This existing warning has had the *this will be a fatal error* text added in this release.

  This warning is now enabled by default, as all `deprecated` category warnings should be.

- *`$*` is no longer supported. Its use will be fatal in Perl 5.30*

  This existing warning has had the *its use will be fatal* text added in this release.

- *$# is no longer supported. Its use will be fatal in Perl 5.30*

  This existing warning has had the *its use will be fatal* text added in this release.

- *Malformed UTF-8 character%s*

  Details as to the exact problem have been added at the end of this message

- *Missing or undefined argument to %s*

  This warning used to warn about `require`, even if it was actually `do` which being executed. It now gets the operation name right.

- NO-BREAK SPACE in a charnames alias definition is deprecated

  This warning has been removed as the behavior is now an error.

- *Odd name/value argument for subroutine '%s'*

  This warning now includes the name of the offending subroutine.

- *Opening dirhandle %s also as a file. This will be a fatal error in Perl 5.28*

  This existing warning has had the *this will be a fatal error* text added in this release.

- *Opening filehandle %s also as a directory. This will be a fatal error in Perl 5.28*

  This existing warning has had the *this will be a fatal error* text added in this release.

- panic: ck_split, type=%u

  panic: pp_split, pm=%p, s=%p

  These panic errors have been removed.

- Passing malformed UTF-8 to "%s" is deprecated

  This warning has been changed to the fatal *Malformed UTF-8 string in "%s"*

- *Setting $/ to a reference to %s as a form of slurp is deprecated, treating as undef. This will be fatal in Perl 5.28*

  This existing warning has had the *this will be fatal* text added in this release.

- *`${^ENCODING}` is no longer supported. Its use will be fatal in Perl 5.28*

  This warning used to be: "Setting `${^ENCODING}` is deprecated".

  The special action of the variable `${^ENCODING}` was formerly used to implement the `encoding` pragma. As of Perl 5.26, rather than being deprecated, assigning to this variable now has no effect except to issue the warning.

- *Too few arguments for subroutine '%s'*

  This warning now includes the name of the offending subroutine.

- *Too many arguments for subroutine '%s'*

  This warning now includes the name of the offending subroutine.

- *Unescaped left brace in regex is deprecated here (and will be fatal in Perl 5.30), passed through in regex; marked by <-- HERE in m/%s/*

  This existing warning has had the *here (and will be fatal...)* text added in this release.

- *Unknown charname '' is deprecated. Its use will be fatal in Perl 5.28*

  This existing warning has had the *its use will be fatal* text added in this release.

- *Use of bare << to mean <<"" is deprecated. Its use will be fatal in Perl 5.28*

  This existing warning has had the *its use will be fatal* text added in this release.

- *Use of code point 0x%s is deprecated; the permissible max is 0x%s. This will be fatal in Perl 5.28*

  This existing warning has had the *this will be fatal* text added in this release.
- *Use of comma-less variable list is deprecated. Its use will be fatal in Perl 5.28*

  This existing warning has had the *its use will be fatal* text added in this release.
- *Use of inherited `AUTOLOAD` for non-method %s() is deprecated. This will be fatal in Perl 5.28*

  This existing warning has had the *this will be fatal* text added in this release.
- *Use of strings with code points over 0xFF as arguments to %s operator is deprecated. This will be a fatal error in Perl 5.28*

  This existing warning has had the *this will be a fatal error* text added in this release.

Utility Changes

c2ph and pstruct

- These old utilities have long since superceded by *h2xs*, and are now gone from the distribution.

Porting/pod_lib.pl

- Removed spurious executable bit.
- Account for the possibility of DOS file endings.

Porting/sync-with-cpan

- Many improvements.

perf/benchmarks

- Tidy file, rename some symbols.

Porting/checkAUTHORS.pl

- Replace obscure character range with `\w`.

t/porting/regen.t

- Try to be more helpful when tests fail.

utils/h2xs.PL

- Avoid infinite loop for enums.

perlbug

- Long lines in the message body are now wrapped at 900 characters, to stay well within the 1000-character limit imposed by SMTP mail transfer agents. This is particularly likely to be important for the list of arguments to *Configure*, which can readily exceed the limit if, for example, it names several non-default installation paths. This change also adds the first unit tests for perlbug. *[perl #128020]*

Configuration and Compilation

- `-Ddefault_inc_excludes_dot` has added, and enabled by default.
- The `dtrace` build process has further changes *[perl #130108]*:
  - If the `-xnolibs` is available, use that so a *dtrace* perl can be built within a FreeBSD jail.
  - On systems that build a *dtrace* object file (FreeBSD, Solaris, and SystemTap's dtrace emulation), copy the input objects to a separate directory and process them there, and use those objects in the link, since `dtrace -G` also modifies these objects.

  - Add *libelf* to the build on FreeBSD 10.x, since *dtrace* adds references to *libelf* symbols.
  - Generate a dummy *dtrace_main.o* if `dtrace -G` fails to build it. A default build on Solaris generates probes from the unused inline functions, while they don't on FreeBSD, which causes `dtrace -G` to fail.

- You can now disable perl's use of the `PERL_HASH_SEED` and `PERL_PERTURB_KEYS` environment variables by configuring perl with `-Accflags=NO_PERL_HASH_ENV`.
- You can now disable perl's use of the `PERL_HASH_SEED_DEBUG` environment variable by configuring perl with `-Accflags=-DNO_PERL_HASH_SEED_DEBUG`.
- *Configure* now zeroes out the alignment bytes when calculating the bytes for 80-bit `NaN` and `Inf` to make builds more reproducible. *[perl #130133]*
- Since v5.18, for testing purposes we have included support for building perl with a variety of non-standard, and non-recommended hash functions. Since we do not recommend the use of these functions, we have removed them and their corresponding build options. Specifically this includes the following build options:

```
PERL_HASH_FUNC_SDBM
PERL_HASH_FUNC_DJB2
PERL_HASH_FUNC_SUPERFAST
PERL_HASH_FUNC_MURMUR3
PERL_HASH_FUNC_ONE_AT_A_TIME
PERL_HASH_FUNC_ONE_AT_A_TIME_OLD
PERL_HASH_FUNC_MURMUR_HASH_64A
PERL_HASH_FUNC_MURMUR_HASH_64B
```

- Remove "Warning: perl appears in your path"

  This install warning is more or less obsolete, since most platforms already **will** have a */usr/bin/perl* or similar provided by the OS.
- Reduce verbosity of `make install.man`

  Previously, two progress messages were emitted for each manpage: one by installman itself, and one by the function in *install_lib.pl* that it calls to actually install the file. Disabling the second of those in each case saves over 750 lines of unhelpful output.
- Cleanup for `clang -Weverything` support. *[perl #129961]*
- *Configure*: signbit scan was assuming too much, stop assuming negative 0.
- Various compiler warnings have been silenced.
- Several smaller changes have been made to remove impediments to compiling under C++11.
- Builds using `USE_PAD_RESET` now work again; this configuration had bit-rotted.
- A probe for `gai_strerror` was added to *Configure* that checks if the `gai_strerror()` routine is available and can be used to translate error codes returned by `getaddrinfo()` into human readable strings.
- *Configure* now aborts if both `-Duselongdouble` and `-Dusequadmath` are requested. *[perl #126203]*
- Fixed a bug in which *Configure* could append `-quadmath` to the archname even if it was already present. *[perl #128538]*
- Clang builds with `-DPERL_GLOBAL_STRUCT` or `-DPERL_GLOBAL_STRUCT_PRIVATE` have been fixed (by disabling Thread Safety Analysis for these configurations).

- *make_ext.pl* no longer updates a module's *pm_to_blib* file when no files require updates. This could cause dependencies, *perlmain.c* in particular, to be rebuilt unnecessarily. *[perl #126710]*
- The output of `perl -V` has been reformatted so that each configuration and compile-time option is now listed one per line, to improve readability.
- *Configure* now builds `miniperl` and `generate_uudmap` if you invoke it with `-Dusecrosscompiler` but not `-Dtargethost=somehost`. This means you can supply your target platform `config.sh`, generate the headers and proceed to build your cross-target perl. *[perl #127234]*
- Perl built with `-Accflags=-DPERL_TRACE_OPS` now only dumps the operator counts when the environment variable `PERL_TRACE_OPS` is set to a non-zero integer. This allows `make test` to pass on such a build.
- When building with GCC 6 and link-time optimization (the `-flto` option to `gcc`), *Configure* was treating all probed symbols as present on the system, regardless of whether they actually exist. This has been fixed. *[perl #128131]*
- The *t/test.pl* library is used for internal testing of Perl itself, and also copied by several CPAN modules. Some of those modules must work on older versions of Perl, so *t/test.pl* must in turn avoid newer Perl features. Compatibility with Perl 5.8 was inadvertently removed some time ago; it has now been restored. *[perl #128052]*
- The build process no longer emits an extra blank line before building each "simple" extension (those with only **.pm* and **.pod* files).

Testing

Tests were added and changed to reflect the other additions and changes in this release. Furthermore, these substantive changes were made:

- A new test script, *comp/parser_run.t*, has been added that is like *comp/parser.t* but with *test.pl* included so that `runperl()` and the like are available for use.
- Tests for locales were erroneously using locales incompatible with Perl.
- Some parts of the test suite that try to exhaustively test edge cases in the regex implementation have been restricted to running for a maximum of five minutes. On slow systems they could otherwise take several hours, without significantly improving our understanding of the correctness of the code under test.
- A new internal facility allows analysing the time taken by the individual tests in Perl's own test suite; see *Porting/harness-timer-report.pl*.
- *t/re/regexp_nonull.t* has been added to test that the regular expression engine can handle scalars that do not have a null byte just past the end of the string.
- A new test script, *t/op/decl-refs.t*, has been added to test the new feature *Declaring a reference to a variable*.
- A new test script, *t/re/keep_tabs.t* has been added to contain tests where `\t` characters should not be expanded into spaces.
- A new test script, *t/re/anyof.t*, has been added to test that the ANYOF nodes generated by bracketed character classes are as expected.
- There is now more extensive testing of the Unicode-related API macros and functions.
- Several of the longer running API test files have been split into multiple test files so that they can be run in parallel.
- *t/harness* now tries really hard not to run tests which are located outside of the Perl source

tree. *[perl #124050]*

- Prevent debugger tests (*lib/perl5db.t*) from failing due to the contents of `$ENV{PERLDB_OPTS}`. *[perl #130445]*

Platform Support

New Platforms

NetBSD/VAX

Perl now compiles under NetBSD on VAX machines. However, it's not possible for that platform to implement floating-point infinities and NaNs compatible with most modern systems, which implement the IEEE-754 floating point standard. The hexadecimal floating point (`0x...p[+-]n` literals, `printf %a`) is not implemented, either. The `make test` passes 98% of tests.

- Test fixes and minor updates.
- Account for lack of `inf`, `nan`, and `-0.0` support.

Platform-Specific Notes

Darwin

- Don't treat `-Dprefix=/usr` as special: instead require an extra option `-Ddarwin_distribution` to produce the same results.
- OS X El Capitan doesn't implement the `clock_gettime()` or `clock_getres()` APIs; emulate them as necessary.
- Deprecated `syscall(2)` on macOS 10.12.

EBCDIC

Several tests have been updated to work (or be skipped) on EBCDIC platforms.

HP-UX

The *Net::Ping* UDP test is now skipped on HP-UX.

Hurd

The hints for Hurd have been improved, enabling malloc wrap and reporting the GNU libc used (previously it was an empty string when reported).

VAX

VAX floating point formats are now supported on NetBSD.

VMS

- The path separator for the `PERL5LIB` and `PERLLIB` environment entries is now a colon (`":"`) when running under a Unix shell. There is no change when running under DCL (it's still `"|"`).
- *configure.com* now recognizes the VSI-branded C compiler and no longer recognizes the "DEC"-branded C compiler (as there hasn't been such a thing for 15 or more years).

Windows

- Support for compiling perl on Windows using Microsoft Visual Studio 2015 (containing Visual C++ 14.0) has been added.

  This version of VC++ includes a completely rewritten C run-time library, some of the changes in which mean that work done to resolve a socket `close()` bug in perl #120091 and perl #118059 is not workable in its current state with this version of VC++. Therefore, we have effectively reverted that bug fix for VS2015 onwards on the basis that being able to build with VS2015 onwards is more important than keeping the bug fix. We may revisit this in the future to attempt to fix the bug again in a way that is compatible with VS2015.

These changes do not affect compilation with GCC or with Visual Studio versions up to and including VS2013, *i.e.*, the bug fix is retained (unchanged) for those compilers.

Note that you may experience compatibility problems if you mix a perl built with GCC or VS <= VS2013 with XS modules built with VS2015, or if you mix a perl built with VS2015 with XS modules built with GCC or VS <= VS2013. Some incompatibility may arise because of the bug fix that has been reverted for VS2015 builds of perl, but there may well be incompatibility anyway because of the rewritten CRT in VS2015 (*e.g.*, see discussion at *http://stackoverflow.com/questions/30412951*).

- It now automatically detects GCC versus Visual C and sets the VC version number on Win32.

Linux

Drop support for Linux *a.out* executable format. Linux has used ELF for over twenty years.

OpenBSD 6

OpenBSD 6 still does not support returning `pid`, `gid`, or `uid` with `SA_SIGINFO`. Make sure to account for it.

FreeBSD

t/uni/overload.t: Skip hanging test on FreeBSD.

DragonFly BSD

DragonFly BSD now has support for `setproctitle()`. *[perl #130068]*.

Internal Changes

- A new API function *`sv_setpv_bufsize()`* allows simultaneously setting the length and the allocated size of the buffer in an SV, growing the buffer if necessary.
- A new API macro *`SvPVCLEAR()`* sets its SV argument to an empty string, like Perl-space `$x = ''`, but with several optimisations.
- Several new macros and functions for dealing with Unicode and UTF-8-encoded strings have been added to the API, as well as some changes in the functionality of existing functions (see *"Unicode Support" in perlapi* for more details):
    - New versions of the API macros like `isALPHA_utf8` and `toLOWER_utf8` have been added, each with the suffix `_safe`, like *`isSPACE_utf8_safe`*. These take an extra parameter, giving an upper limit of how far into the string it is safe to read. Using the old versions could cause attempts to read beyond the end of the input buffer if the UTF-8 is not well-formed, and their use now raises a deprecation warning. Details are at *"Character classification" in perlapi*.
    - Macros like *`isALPHA_utf8`* and *`toLOWER_utf8`* now die if they detect that their input UTF-8 is malformed. A deprecation warning had been issued since Perl 5.18.
    - Several new macros for analysing the validity of utf8 sequences. These are:

      `UTF8_GOT_ABOVE_31_BIT` *`UTF8_GOT_CONTINUATION`* *`UTF8_GOT_EMPTY`* *`UTF8_GOT_LONG`* *`UTF8_GOT_NONCHAR`* *`UTF8_GOT_NON_CONTINUATION`* *`UTF8_GOT_OVERFLOW`* *`UTF8_GOT_SHORT`* *`UTF8_GOT_SUPER`* *`UTF8_GOT_SURROGATE`* *`UTF8_IS_INVARIANT`* *`UTF8_IS_NONCHAR`* *`UTF8_IS_SUPER`* *`UTF8_IS_SURROGATE`* *`UVCHR_IS_INVARIANT`* *`isUTF8_CHAR_flags`* *`isSTRICT_UTF8_CHAR`* *`isC9_STRICT_UTF8_CHAR`*
    - Functions that are all extensions of the `is_utf8_string_*()` functions, that apply

various restrictions to the UTF-8 recognized as valid:

is_strict_utf8_string, *is_strict_utf8_string_loc*, *is_strict_utf8_string_loclen*,

is_c9strict_utf8_string, *is_c9strict_utf8_string_loc*, *is_c9strict_utf8_string_loclen*,

is_utf8_string_flags, *is_utf8_string_loc_flags*, *is_utf8_string_loclen_flags*,

is_utf8_fixed_width_buf_flags, *is_utf8_fixed_width_buf_loc_flags*, *is_utf8_fixed_width_buf_loclen_flags*.

is_utf8_invariant_string. *is_utf8_valid_partial_char*. *is_utf8_valid_partial_char_flags*.

- The functions *utf8n_to_uvchr* and its derivatives have had several changes of behaviour.

  Calling them, while passing a string length of 0 is now asserted against in DEBUGGING builds, and otherwise, returns the Unicode REPLACEMENT CHARACTER. If you have nothing to decode, you shouldn't call the decode function.

  They now return the Unicode REPLACEMENT CHARACTER if called with UTF-8 that has the overlong malformation and that malformation is allowed by the input parameters. This malformation is where the UTF-8 looks valid syntactically, but there is a shorter sequence that yields the same code point. This has been forbidden since Unicode version 3.1.

  They now accept an input flag to allow the overflow malformation. This malformation is when the UTF-8 may be syntactically valid, but the code point it represents is not capable of being represented in the word length on the platform. What "allowed" means, in this case, is that the function doesn't return an error, and it advances the parse pointer to beyond the UTF-8 in question, but it returns the Unicode REPLACEMENT CHARACTER as the value of the code point (since the real value is not representable).

  They no longer abandon searching for other malformations when the first one is encountered. A call to one of these functions thus can generate multiple diagnostics, instead of just one.

- *valid_utf8_to_uvchr()* has been added to the API (although it was present in core earlier). Like `utf8_to_uvchr_buf()`, but assumes that the next character is well-formed. Use with caution.

- A new function, *utf8n_to_uvchr_error*, has been added for use by modules that need to know the details of UTF-8 malformations beyond pass/fail. Previously, the only ways to know why a sequence was ill-formed was to capture and parse the generated diagnostics or to do your own analysis.

- There is now a safer version of utf8_hop(), called *utf8_hop_safe()*. Unlike utf8_hop(), utf8_hop_safe() won't navigate before the beginning or after the end of the supplied buffer.

- Two new functions, *utf8_hop_forward()* and *utf8_hop_back()* are similar to `utf8_hop_safe()` but are for when you know which direction you wish to travel.

- Two new macros which return useful utf8 byte sequences:

  BOM_UTF8

  REPLACEMENT_CHARACTER_UTF8

- Perl is now built with the `PERL_OP_PARENT` compiler define enabled by default. To disable it, use the `PERL_NO_OP_PARENT` compiler define. This flag alters how the `op_sibling` field is used in `OP` structures, and has been available optionally since perl 5.22.

See *"Internal Changes" in perl5220delta* for more details of what this build option does.

- Three new ops, `OP_ARGELEM`, `OP_ARGDEFELEM`, and `OP_ARGCHECK` have been added. These are intended principally to implement the individual elements of a subroutine signature, plus any overall checking required.
- The `OP_PUSHRE` op has been eliminated and the `OP_SPLIT` op has been changed from class `LISTOP` to `PMOP`.

  Formerly the first child of a split would be a `pushre`, which would have the `split`'s regex attached to it. Now the regex is attached directly to the `split` op, and the `pushre` has been eliminated.
- The *`op_class()`* API function has been added. This is like the existing `OP_CLASS()` macro, but can more accurately determine what struct an op has been allocated as. For example `OP_CLASS()` might return `OA_BASEOP_OR_UNOP` indicating that ops of this type are usually allocated as an `OP` or `UNOP`; while `op_class()` will return `OPclass_BASEOP` or `OPclass_UNOP` as appropriate.
- All parts of the internals now agree that the `sassign` op is a `BINOP`; previously it was listed as a `BASEOP` in *regen/opcodes*, which meant that several parts of the internals had to be special-cased to accommodate it. This oddity's original motivation was to handle code like `$x ||= 1`; that is now handled in a simpler way.
- The output format of the *`op_dump()`* function (as used by `perl -Dx`) has changed: it now displays an "ASCII-art" tree structure, and shows more low-level details about each op, such as its address and class.
- The `PADOFFSET` type has changed from being unsigned to signed, and several pad-related variables such as `PL_padix` have changed from being of type `I32` to type `PADOFFSET`.
- The `DEBUGGING`-mode output for regex compilation and execution has been enhanced.
- Several obscure SV flags have been eliminated, sometimes along with the macros which manipulate them: `SVpbm_VALID`, `SVpbm_TAIL`, `SvTAIL_on`, `SvTAIL_off`, `SVrepl_EVAL`, `SvEVALED`.
- An OP `op_private` flag has been eliminated: `OPpRUNTIME`. This used to often get set on `PMOP` ops, but had become meaningless over time.

Selected Bug Fixes

- Perl no longer panics when switching into some locales on machines with buggy `strxfrm()` implementations in their *libc*. *[perl #121734]*
- `$-{$name}` would leak an `AV` on each access if the regular expression had no named captures. The same applies to access to any hash tied with *Tie::Hash::NamedCapture* and `all => 1`. *[perl #130822]*
- Attempting to use the deprecated variable `$#` as the object in an indirect object method call could cause a heap use after free or buffer overflow. *[perl #129274]*
- When checking for an indirect object method call, in some rare cases the parser could reallocate the line buffer but then continue to use pointers to the old buffer. *[perl #129190]*
- Supplying a glob as the format argument to *`formline`* would cause an assertion failure. *[perl #130722]*
- Code like `$value1 =~ qr/.../ ~~ $value2` would have the match converted into a `qr//` operator, leaving extra elements on the stack to confuse any surrounding expression.

[perl #130705]

- Since v5.24 in some obscure cases, a regex which included code blocks from multiple sources (*e.g.*, via embedded via `qr//` objects) could end up with the wrong current pad and crash or give weird results. *[perl #129881]*
- Occasionally `local()`s in a code block within a patterns weren't being undone when the pattern matching backtracked over the code block. *[perl #126697]*
- Using `substr()` to modify a magic variable could access freed memory in some cases. *[perl #129340]*
- Under `use utf8`, the entire source code is now checked for being UTF-8 well formed, not just quoted strings as before. *[perl #126310]*.
- The range operator `".."` on strings now handles its arguments correctly when in the scope of the *`unicode_strings`* feature. The previous behaviour was sufficiently unexpected that we believe no correct program could have made use of it.
- The `split` operator did not ensure enough space was allocated for its return value in scalar context. It could then write a single pointer immediately beyond the end of the memory block allocated for the stack. *[perl #130262]*
- Using a large code point with the `"W"` pack template character with the current output position aligned at just the right point could cause a write of a single zero byte immediately beyond the end of an allocated buffer. *[perl #129149]*
- Supplying a format's picture argument as part of the format argument list where the picture specifies modifying the argument could cause an access to the new freed compiled form.at. *[perl #129125]*
- The *sort()* operator's built-in numeric comparison function didn't handle large integers that weren't exactly representable by a double. This now uses the same code used to implement the <=> operator. *[perl #130335]*
- Fix issues with `/(?{ ... <<EOF })/` that broke *Method::Signatures*. *[perl #130398]*
- Fixed an assertion failure with `chop` and `chomp`, which could be triggered by `chop(@x =~ tr/1/1/)`. *[perl #130198]*.
- Fixed a comment skipping error in patterns under `/x`; it could stop skipping a byte early, which could be in the middle of a UTF-8 character. *[perl #130495]*.
- *perldb* now ignores */dev/tty* on non-Unix systems. *[perl #113960]*;
- Fix assertion failure for `{}->$x` when `$x` isn't defined. *[perl #130496]*.
- Fix an assertion error which could be triggered when a lookahead string in patterns exceeded a minimum length. *[perl #130522]*.
- Only warn once per literal number about a misplaced `"_"`. *[perl #70878]*.
- The `tr///` parse code could be looking at uninitialized data after a perse error. *[perl #129342]*.
- In a pattern match, a back-reference (`\1`) to an unmatched capture could read back beyond the start of the string being matched. *[perl #129377]*.
- `use re 'strict'` is supposed to warn if you use a range (such as `/(?[ [ X-Y ] ])/`) whose start and end digit aren't from the same group of 10. It didn't do that for five groups of mathematical digits starting at `U+1D7E`.
- A sub containing a "forward" declaration with the same name (*e.g.*, `sub c { sub c; }`)

could sometimes crash or loop infinitely. *[perl #129090]*

- A crash in executing a regex with a non-anchored UTF-8 substring against a target string that also used UTF-8 has been fixed. *[perl #129350]*
- Previously, a shebang line like `#!perl -i u` could be erroneously interpreted as requesting the `-u` option. This has been fixed. *[perl #129336]*
- The regex engine was previously producing incorrect results in some rare situations when backtracking past an alternation that matches only one thing; this showed up as capture buffers (`$1`, `$2`, *etc.*) erroneously containing data from regex execution paths that weren't actually executed for the final match. *[perl #129897]*
- Certain regexes making use of the experimental `regex_sets` feature could trigger an assertion failure. This has been fixed. *[perl #129322]*
- Invalid assignments to a reference constructor (*e.g.*, `\eval=time`) could sometimes crash in addition to giving a syntax error. *[perl #125679]*
- The parser could sometimes crash if a bareword came after `evalbytes`. *[perl #129196]*
- Autoloading via a method call would warn erroneously ("Use of inherited AUTOLOAD for non-method") if there was a stub present in the package into which the invocant had been blessed. The warning is no longer emitted in such circumstances. *[perl #47047]*
- The use of `splice` on arrays with non-existent elements could cause other operators to crash. *[perl #129164]*
- A possible buffer overrun when a pattern contains a fixed utf8 substring. *[perl #129012]*
- Fixed two possible use-after-free bugs in perl's lexer. *[perl #129069]*
- Fixed a crash with `s///l` where it thought it was dealing with UTF-8 when it wasn't. *[perl #129038]*
- Fixed a place where the regex parser was not setting the syntax error correctly on a syntactically incorrect pattern. *[perl #129122]*
- The `&.` operator (and the `"&"` operator, when it treats its arguments as strings) were failing to append a trailing null byte if at least one string was marked as utf8 internally. Many code paths (system calls, regexp compilation) still expect there to be a null byte in the string buffer just past the end of the logical string. An assertion failure was the result. *[perl #129287]*
- Avoid a heap-after-use error in the parser when creating an error messge for a syntactically invalid heredoc. *[perl #128988]*
- Fix a segfault when run with `-DC` options on DEBUGGING builds. *[perl #129106]*
- Fixed the parser error handling in subroutine attributes for an '`:attr(foo`' that does not have an ending '`")"`'.
- Fix the perl lexer to correctly handle a backslash as the last char in quoted-string context. This actually fixed two bugs, *[perl #129064]* and *[perl #129176]*.
- In the API function `gv_fetchmethod_pvn_flags`, rework separator parsing to prevent possible string overrun with an invalid `len` argument. *[perl #129267]*
- Problems with in-place array sorts: code like `@a = sort { ... } @a`, where the source and destination of the sort are the same plain array, are optimised to do less copying around. Two side-effects of this optimisation were that the contents of `@a` as seen by sort routines were partially sorted; and under some circumstances accessing `@a` during the sort could crash the interpreter. Both these issues have been fixed, and Sort functions see the original value of

`@a`. *[perl #128340]*

- Non-ASCII string delimiters are now reported correctly in error messages for unterminated strings. *[perl #128701]*
- `pack("p", ...)` used to emit its warning ("Attempt to pack pointer to temporary value") erroneously in some cases, but has been fixed.
- `@DB::args` is now exempt from "used once" warnings. The warnings only occurred under **-w**, because *warnings.pm* itself uses `@DB::args` multiple times.
- The use of built-in arrays or hash slices in a double-quoted string no longer issues a warning ("Possible unintended interpolation...") if the variable has not been mentioned before. This affected code like `qq|@DB::args|` and `qq|@SIG{'CHLD', 'HUP'}|`. (The special variables `@-` and `@+` were already exempt from the warning.)
- `gethostent` and similar functions now perform a null check internally, to avoid crashing with the torsocks library. This was a regression from v5.22. *[perl #128740]*
- `defined *{'!'}`, `defined *{'['}`, and `defined *{'-'}` no longer leak memory if the typeglob in question has never been accessed before.
- Mentioning the same constant twice in a row (which is a syntax error) no longer fails an assertion under debugging builds. This was a regression from v5.20. *[perl #126482]*
- Many issues relating to `printf "%a"` of hexadecimal floating point were fixed. In addition, the "subnormals" (formerly known as "denormals") floating point numbers are now supported both with the plain IEEE 754 floating point numbers (64-bit or 128-bit) and the x86 80-bit "extended precision". Note that subnormal hexadecimal floating point literals will give a warning about "exponent underflow". *[perl #128843] [perl #128889] [perl #128890] [perl #128893] [perl #128909] [perl #128919]*
- A regression in v5.24 with `tr/\N{U+...}/foo/` when the code point was between 128 and 255 has been fixed. *[perl #128734]*.
- Use of a string delimiter whose code point is above 2**31 now works correctly on platforms that allow this. Previously, certain characters, due to truncation, would be confused with other delimiter characters with special meaning (such as `"?"` in `m?...?`), resulting in inconsistent behaviour. Note that this is non-portable, and is based on Perl's extension to UTF-8, and is probably not displayable nor enterable by any editor. *[perl #128738]*
- `@{x` followed by a newline where `"x"` represents a control or non-ASCII character no longer produces a garbled syntax error message or a crash. *[perl #128951]*
- An assertion failure with `%: = 0` has been fixed. *[perl #128238]*
- In Perl 5.18, the parsing of `"$foo::$bar"` was accidentally changed, such that it would be treated as `$foo."::".$bar`. The previous behavior, which was to parse it as `$foo:: . $bar`, has been restored. *[perl #128478]*
- Since Perl 5.20, line numbers have been off by one when perl is invoked with the **-x** switch. This has been fixed. *[perl #128508]*
- Vivifying a subroutine stub in a deleted stash (*e.g.*, `delete $My::{"Foo::"}; \&My::Foo::foo`) no longer crashes. It had begun crashing in Perl 5.18. *[perl #128532]*
- Some obscure cases of subroutines and file handles being freed at the same time could result in crashes, but have been fixed. The crash was introduced in Perl 5.22. *[perl #128597]*
- Code that looks for a variable name associated with an uninitialized value could cause an assertion failure in cases where magic is involved, such as `$ISA[0][0]`. This has now been fixed. *[perl #128253]*

- A crash caused by code generating the warning "Subroutine STASH::NAME redefined" in cases such as `sub P::f{} undef *P::; *P::f =sub{};` has been fixed. In these cases, where the STASH is missing, the warning will now appear as "Subroutine NAME redefined". *[perl #128257]*

- Fixed an assertion triggered by some code that handles deprecated behavior in formats, *e.g.*, in cases like this:

  ```
      format STDOUT =
      @
      0"$x"
  ```

  [perl #128255]

- A possible divide by zero in string transformation code on Windows has been avoided, fixing a crash when collating an empty string. *[perl #128618]*

- Some regular expression parsing glitches could lead to assertion failures with regular expressions such as `/(?<=/` and `/(?<!/`. This has now been fixed. *[perl #128170]*

- `until ($x = 1) { ... }` and `... until $x = 1` now properly warn when syntax warnings are enabled. *[perl #127333]*

- socket() now leaves the error code returned by the system in `$!` on failure. *[perl #128316]*

- Assignment variants of any bitwise ops under the `bitwise` feature would crash if the left-hand side was an array or hash. *[perl #128204]*

- `require` followed by a single colon (as in `foo() ? require : ...` is now parsed correctly as `require` with implicit $_, rather than `require ""`. *[perl #128307]*

- Scalar `keys %hash` can now be assigned to consistently in all scalar lvalue contexts. Previously it worked for some contexts but not others.

- List assignment to `vec` or `substr` with an array or hash for its first argument used to result in crashes or "Can't coerce" error messages at run time, unlike scalar assignment, which would give an error at compile time. List assignment now gives a compile-time error, too. *[perl #128260]*

- Expressions containing an `&&` or `||` operator (or their synonyms `and` and `or`) were being compiled incorrectly in some cases. If the left-hand side consisted of either a negated bareword constant or a negated `do {}` block containing a constant expression, and the right-hand side consisted of a negated non-foldable expression, one of the negations was effectively ignored. The same was true of `if` and `unless` statement modifiers, though with the left-hand and right-hand sides swapped. This long-standing bug has now been fixed. *[perl #127952]*

- `reset` with an argument no longer crashes when encountering stash entries other than globs. *[perl #128106]*

- Assignment of hashes to, and deletion of, typeglobs named `*::::::` no longer causes crashes. *[perl #128086]*

- Perl wasn't correctly handling true/false values in the LHS of a list assign; specifically the truth values returned by boolean operators. This could trigger an assertion failure in something like the following:

  ```
      for ($x > $y) {
          ($_, ...) = (...); # here $_ is aliased to a truth value
      }
  ```

  This was a regression from v5.24. *[perl #129991]*

- Assertion failure with user-defined Unicode-like properties. *[perl #130010]*
- Fix error message for unclosed `\N{` in a regex. An unclosed `\N{` could give the wrong error message: `"\N{NAME} must be resolved by the lexer"`.
- List assignment in list context where the LHS contained aggregates and where there were not enough RHS elements, used to skip scalar lvalues. Previously, `(($a,$b,@c,$d) = (1))` in list context returned `($a)`; now it returns `($a,$b,$d)`. `(($a,$b,$c) = (1))` is unchanged: it still returns `($a,$b,$c)`. This can be seen in the following:

  ```
  sub inc { $_++ for @_ }
  inc(($a,$b,@c,$d) = (10))
  ```

  Formerly, the values of `($a,$b,$d)` would be left as `(11,undef,undef)`; now they are `(11,1,1)`.
- Code like this: `/(?{ s!!! })/` could trigger infinite recursion on the C stack (not the normal perl stack) when the last successful pattern in scope is itself. We avoid the segfault by simply forbidding the use of the empty pattern when it would resolve to the currently executing pattern. *[perl #129903]*
- Avoid reading beyond the end of the line buffer in perl's lexer when there's a short UTF-8 character at the end. *[perl #128997]*
- Alternations in regular expressions were sometimes failing to match a utf8 string against a utf8 alternate. *[perl #129950]*
- Make `do "a\0b"` fail silently (and return `undef` and set `$!`) instead of throwing an error. *[perl #129928]*
- `chdir` with no argument didn't ensure that there was stack space available for returning its result. *[perl #129130]*
- All error messages related to `do` now refer to `do`; some formerly claimed to be from `require` instead.
- Executing `undef $x` where `$x` is tied or magical no longer incorrectly blames the variable for an uninitialized-value warning encountered by the tied/magical code.
- Code like `$x = $x . "a"` was incorrectly failing to yield a *use of uninitialized value* warning when `$x` was a lexical variable with an undefined value. That has now been fixed. *[perl #127877]*
- `undef *_; shift` or `undef *_; pop` inside a subroutine, with no argument to `shift` or `pop`, began crashing in Perl 5.14, but has now been fixed.
- `"string$scalar->$*"` now correctly prefers concatenation overloading to string overloading if `$scalar->$*` returns an overloaded object, bringing it into consistency with `$$scalar`.
- `/@0{0*->@*/*0` and similar contortions used to crash, but no longer do, but merely produce a syntax error. *[perl #128171]*
- `do` or `require` with an argument which is a reference or typeglob which, when stringified, contains a null character, started crashing in Perl 5.20, but has now been fixed. *[perl #128182]*
- Improve the error message for a missing `tie()` package/method. This brings the error messages in line with the ones used for normal method calls.
- Parsing bad POSIX charclasses no longer leaks memory. *[perl #128313]*

Known Problems

- G++ 6 handles subnormal (denormal) floating point values differently than gcc 6 or g++ 5 resulting in "flush-to-zero". The end result is that if you specify very small values using the hexadecimal floating point format, like `0x1.fffffffffffffp-1022`, they become zeros. *[perl #131388]*

Errata From Previous Releases

- Fixed issues with recursive regexes. The behavior was fixed in Perl 5.24. *[perl #126182]*

Obituary

Jon Portnoy (AVENJ), a prolific Perl author and admired Gentoo community member, has passed away on August 10, 2016. He will be remembered and missed by all those who he came in contact with, and enriched with his intellect, wit, and spirit.

It is with great sadness that we also note Kip Hampton's passing. Probably best known as the author of the Perl & XML column on XML.com, he was a core contributor to AxKit, an XML server platform that became an Apache Foundation project. He was a frequent speaker in the early days at OSCON, and most recently at YAPC::NA in Madison. He was frequently on irc.perl.org as ubu, generally in the #axkit-dahut community, the group responsible for YAPC::NA Asheville in 2011.

Kip and his constant contributions to the community will be greatly missed.

Acknowledgements

Perl 5.26.0 represents approximately 13 months of development since Perl 5.24.0 and contains approximately 360,000 lines of changes across 2,600 files from 86 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 230,000 lines of changes to 1,800 .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 5.26.0:

Aaron Crane, Abigail, Ã†var ArnfjÃ¶rÃ° Bjarmason, Alex Vandiver, Andreas KÃ¶nig, Andreas Voegele, Andrew Fresh, Andy Lester, Aristotle Pagaltzis, Chad Granum, Chase Whitener, Chris 'BinGOs' Williams, Chris Lamb, Christian Hansen, Christian Millour, Colin Newell, Craig A. Berry, Dagfinn Ilmari MannsÃ¥ker, Dan Collins, Daniel Dragan, Dave Cross, Dave Rolsky, David Golden, David H. Gutteridge, David Mitchell, Dominic Hargreaves, Doug Bell, E. Choroba, Ed Avis, Father Chrysostomos, FranÃ§ois Perrad, Hauke D, H.Merijn Brand, Hugo van der Sanden, Ivan Pozdeev, James E Keenan, James Raspass, Jarkko Hietaniemi, Jerry D. Hedden, Jim Cromie, J. Nick Koston, John Lightsey, Karen Etheridge, Karl Williamson, Leon Timmermans, Lukas Mai, Matthew Horsfall, Maxwell Carey, Misty De Meo, Neil Bowers, Nicholas Clark, Nicolas R., Niko Tyni, Pali, Paul Marquess, Peter Avalos, Petr PÃsaÅ™, Pino Toscano, Rafael Garcia-Suarez, Reini Urban, Renee Baecker, Ricardo Signes, Richard Levitte, Rick Delaney, Salvador FandiÃ±o, Samuel Thibault, Sawyer X, SÃ©bastien Aperghis-Tramoni, Sergey Aleynikov, Shlomi Fish, Smylers, Stefan Seifert, Steffen MÃ¼ller, Stevan Little, Steve Hay, Steven Humphrey, Sullivan Beck, Theo Buehler, Thomas Sibley, Todd Rinaldo, Tomasz Konojacki, Tony Cook, Unicode Consortium, Yaroslav Kuzmin, Yves Orton, Zefram.

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 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 see *"SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec* for details of how to report the issue.

Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
    perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

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.