NAME

perl5140delta - what is new for perl v5.14.0

DESCRIPTION

This document describes differences between the 5.12.0 release and the 5.14.0 release.

If you are upgrading from an earlier release such as 5.10.0, first read *perl5120delta*, which describes differences between 5.10.0 and 5.12.0.

Some of the bug fixes in this release have been backported to subsequent releases of 5.12.x. Those are indicated with the 5.12.x version in parentheses.

Notice

As described in *perlpolicy*, the release of Perl 5.14.0 marks the official end of support for Perl 5.10. Users of Perl 5.10 or earlier should consider upgrading to a more recent release of Perl.

Core Enhancements

Unicode

Unicode Version 6.0 is now supported (mostly)

Perl comes with the Unicode 6.0 data base updated with *Corrigendum #8*, with one exception noted below. See *http://unicode.org/versions/Unicode6.0.0/* for details on the new release. Perl does not support any Unicode provisional properties, including the new ones for this release.

Unicode 6.0 has chosen to use the name `BELL` for the character at U+1F514, which is a symbol that looks like a bell, and is used in Japanese cell phones. This conflicts with the long-standing Perl usage of having `BELL` mean the ASCII `BEL` character, U+0007. In Perl 5.14, `\N{BELL}` continues to mean U+0007, but its use generates a deprecation warning message unless such warnings are turned off. The new name for U+0007 in Perl is `ALERT`, which corresponds nicely with the existing shorthand sequence for it, `"\a"`. `\N{BEL}` means U+0007, with no warning given. The character at U+1F514 has no name in 5.14, but can be referred to by `\N{U+1F514}`. In Perl 5.16, `\N{BELL}` will refer to U+1F514; all code that uses `\N{BELL}` should be converted to use `\N{ALERT}`, `\N{BEL}`, or `"\a"` before upgrading.

Full functionality for use feature 'unicode_strings'

This release provides full functionality for `use feature 'unicode_strings'`. Under its scope, all string operations executed and regular expressions compiled (even if executed outside its scope) have Unicode semantics. See *"the 'unicode_strings' feature" in feature*. However, see *Inverted bracketed character classes and multi-character folds*, below.

This feature avoids most forms of the "Unicode Bug" (see *"The "Unicode Bug"" in perlunicode* for details). If there is any possibility that your code will process Unicode strings, you are *strongly* encouraged to use this subpragma to avoid nasty surprises.

\N{NAME} and charnames enhancements

- `\N{`*NAME*`}` and `charnames::vianame` now know about the abbreviated character names listed by Unicode, such as NBSP, SHY, LRO, ZWJ, etc.; all customary abbreviations for the C0 and C1 control characters (such as ACK, BEL, CAN, etc.); and a few new variants of some C1 full names that are in common usage.
- Unicode has several *named character sequences*, in which particular sequences of code points are given names. `\N{`*NAME*`}` now recognizes these.
- `\N{`*NAME*`}`, `charnames::vianame`, and `charnames::viacode` now know about every character in Unicode. In earlier releases of Perl, they didn't know about the Hangul syllables nor several CJK (Chinese/Japanese/Korean) characters.
- It is now possible to override Perl's abbreviations with your own custom aliases.

- You can now create a custom alias of the ordinal of a character, known by `\N{NAME}`, `charnames::vianame()`, and `charnames::viacode()`. Previously, aliases had to be to official Unicode character names. This made it impossible to create an alias for unnamed code points, such as those reserved for private use.

- The new function charnames::string_vianame() is a run-time version of `\N{NAME}`, returning the string of characters whose Unicode name is its parameter. It can handle Unicode named character sequences, whereas the pre-existing charnames::vianame() cannot, as the latter returns a single code point.

See *charnames* for details on all these changes.

New warnings categories for problematic (non-)Unicode code points.

Three new warnings subcategories of "utf8" have been added. These allow you to turn off some "utf8" warnings, while allowing other warnings to remain on. The three categories are: `surrogate` when UTF-16 surrogates are encountered; `nonchar` when Unicode non-character code points are encountered; and `non_unicode` when code points above the legal Unicode maximum of 0x10FFFF are encountered.

Any unsigned value can be encoded as a character

With this release, Perl is adopting a model that any unsigned value can be treated as a code point and encoded internally (as utf8) without warnings, not just the code points that are legal in Unicode. However, unless utf8 or the corresponding sub-category (see previous item) of lexical warnings have been explicitly turned off, outputting or executing a Unicode-defined operation such as upper-casing on such a code point generates a warning. Attempting to input these using strict rules (such as with the `:encoding(UTF-8)` layer) will continue to fail. Prior to this release, handling was inconsistent and in places, incorrect.

Unicode non-characters, some of which previously were erroneously considered illegal in places by Perl, contrary to the Unicode Standard, are now always legal internally. Inputting or outputting them works the same as with the non-legal Unicode code points, because the Unicode Standard says they are (only) illegal for "open interchange".

Unicode database files not installed

The Unicode database files are no longer installed with Perl. This doesn't affect any functionality in Perl and saves significant disk space. If you need these files, you can download them from *http://www.unicode.org/Public/zipped/6.0.0/*.

Regular Expressions

(?^...) construct signifies default modifiers

An ASCII caret `"^"` immediately following a `"(?"` in a regular expression now means that the subexpression does not inherit surrounding modifiers such as `/i`, but reverts to the Perl defaults. Any modifiers following the caret override the defaults.

Stringification of regular expressions now uses this notation. For example, `qr/hlagh/i` would previously be stringified as `(?i-xsm:hlagh)`, but now it's stringified as `(?^i:hlagh)`.

The main purpose of this change is to allow tests that rely on the stringification *not* to have to change whenever new modifiers are added. See *"Extended Patterns" in perlre*.

This change is likely to break code that compares stringified regular expressions with fixed strings containing `?-xism`.

/d, /l, /u, and /a modifiers

Four new regular expression modifiers have been added. These are mutually exclusive: one only can be turned on at a time.

- The `/l` modifier says to compile the regular expression as if it were in the scope of `use`

`locale`, even if it is not.

- The `/u` modifier says to compile the regular expression as if it were in the scope of a `use feature 'unicode_strings'` pragma.
- The `/d` (default) modifier is used to override any `use locale` and `use feature 'unicode_strings'` pragmas in effect at the time of compiling the regular expression.
- The `/a` regular expression modifier restricts `\s`, `\d` and `\w` and the POSIX (`[[:posix:]]`) character classes to the ASCII range. Their complements and `\b` and `\B` are correspondingly affected. Otherwise, `/a` behaves like the `/u` modifier, in that case-insensitive matching uses Unicode semantics.

  If the `/a` modifier is repeated, then additionally in case-insensitive matching, no ASCII character can match a non-ASCII character. For example,

```
"k"     =~ /\N{KELVIN SIGN}/ai
"\xDF" =~ /ss/ai
```

  match but

```
"k"    =~ /\N{KELVIN SIGN}/aai
"\xDF" =~ /ss/aai
```

  do not match.

See *"Modifiers" in perlre* for more detail.

Non-destructive substitution

The substitution (`s///`) and transliteration (`y///`) operators now support an `/r` option that copies the input variable, carries out the substitution on the copy, and returns the result. The original remains unmodified.

```
my $old = "cat";
my $new = $old =~ s/cat/dog/r;
# $old is "cat" and $new is "dog"
```

This is particularly useful with `map`. See *perlop* for more examples.

Re-entrant regular expression engine

It is now safe to use regular expressions within `(?{...})` and `(??{...})` code blocks inside regular expressions.

These blocks are still experimental, however, and still have problems with lexical (`my`) variables and abnormal exiting.

use re '/flags'

The `re` pragma now has the ability to turn on regular expression flags till the end of the lexical scope:

```
use re "/x";
"foo" =~ / (.+) /;  # /x implied
```

See *"'/flags' mode" in re* for details.

\o{...} for octals

There is a new octal escape sequence, `"\o"`, in doublequote-like contexts. This construct allows large octal ordinals beyond the current max of 0777 to be represented. It also allows you to specify a character in octal which can safely be concatenated with other regex snippets and which won't be confused with being a backreference to a regex capture group. See *"Capture groups" in perlre*.

Add \p{Titlecase} as a synonym for \p{Title}

This synonym is added for symmetry with the Unicode property names `\p{Uppercase}` and `\p{Lowercase}`.

Regular expression debugging output improvement

Regular expression debugging output (turned on by `use re 'debug'`) now uses hexadecimal when escaping non-ASCII characters, instead of octal.

Return value of delete $+{...}

Custom regular expression engines can now determine the return value of `delete` on an entry of `%+` or `%-`.

Syntactical Enhancements

Array and hash container functions accept references

Warning: This feature is considered experimental, as the exact behaviour may change in a future version of Perl.

All builtin functions that operate directly on array or hash containers now also accept unblessed hard references to arrays or hashes:

```
|-----------------------------+---------------------------|
| Traditional syntax          | Terse syntax              |
|-----------------------------+---------------------------|
| push @$arrayref, @stuff     | push $arrayref, @stuff    |
| unshift @$arrayref, @stuff  | unshift $arrayref, @stuff |
| pop @$arrayref              | pop $arrayref             |
| shift @$arrayref            | shift $arrayref           |
| splice @$arrayref, 0, 2     | splice $arrayref, 0, 2    |
| keys %$hashref              | keys $hashref             |
| keys @$arrayref             | keys $arrayref            |
| values %$hashref            | values $hashref           |
| values @$arrayref           | values $arrayref          |
| ($k,$v) = each %$hashref    | ($k,$v) = each $hashref   |
| ($k,$v) = each @$arrayref   | ($k,$v) = each $arrayref  |
|-----------------------------+---------------------------|
```

This allows these builtin functions to act on long dereferencing chains or on the return value of subroutines without needing to wrap them in `@{}` or `%{}`:

```
push @{$obj->tags}, $new_tag;  # old way
push $obj->tags,    $new_tag;  # new way

for ( keys %{$hoh->{genres}{artists}} ) {...} # old way
for ( keys $hoh->{genres}{artists}    ) {...} # new way
```

Single term prototype

The + prototype is a special alternative to $ that acts like `\[@%]` when given a literal array or hash variable, but will otherwise force scalar context on the argument. See *"Prototypes" in perlsub*.

package block syntax

A package declaration can now contain a code block, in which case the declaration is in scope inside that block only. So `package Foo { ... }` is precisely equivalent to `{ package Foo; ... }`. It also works with a version number in the declaration, as in `package Foo 1.2 { ... }`, which is its most attractive feature. See *perlfunc*.

Statement labels can appear in more places

Statement labels can now occur before any type of statement or declaration, such as `package`.

Stacked labels

Multiple statement labels can now appear before a single statement.

Uppercase X/B allowed in hexadecimal/binary literals

Literals may now use either upper case `0X...` or `0B...` prefixes, in addition to the already supported `0x...` and `0b...` syntax [perl #76296].

C, Ruby, Python, and PHP already support this syntax, and it makes Perl more internally consistent: a round-trip with `eval sprintf "%#X", 0x10` now returns `16`, just like `eval sprintf "%#x", 0x10`.

Overridable tie functions

`tie`, `tied` and `untie` can now be overridden [perl #75902].

Exception Handling

To make them more reliable and consistent, several changes have been made to how `die`, `warn`, and `$@` behave.

- When an exception is thrown inside an `eval`, the exception is no longer at risk of being clobbered by destructor code running during unwinding. Previously, the exception was written into `$@` early in the throwing process, and would be overwritten if `eval` was used internally in the destructor for an object that had to be freed while exiting from the outer `eval`. Now the exception is written into `$@` last thing before exiting the outer `eval`, so the code running immediately thereafter can rely on the value in `$@` correctly corresponding to that `eval`. (`$@` is still also set before exiting the `eval`, for the sake of destructors that rely on this.)

  Likewise, a `local $@` inside an `eval` no longer clobbers any exception thrown in its scope. Previously, the restoration of `$@` upon unwinding would overwrite any exception being thrown. Now the exception gets to the `eval` anyway. So `local $@` is safe before a `die`.

  Exceptions thrown from object destructors no longer modify the `$@` of the surrounding context. (If the surrounding context was exception unwinding, this used to be another way to clobber the exception being thrown.) Previously such an exception was sometimes emitted as a warning, and then either was string-appended to the surrounding `$@` or completely replaced the surrounding `$@`, depending on whether that exception and the surrounding `$@` were strings or objects. Now, an exception in this situation is always emitted as a warning, leaving the surrounding `$@` untouched. In addition to object destructors, this also affects any function call run by XS code using the `G_KEEPERR` flag.

- Warnings for `warn` can now be objects in the same way as exceptions for `die`. If an object-based warning gets the default handling of writing to standard error, it is stringified as before with the filename and line number appended. But a `$SIG{__WARN__}` handler now receives an object-based warning as an object, where previously it was passed the result of stringifying the object.

Other Enhancements

Assignment to $0 sets the legacy process name with prctl() on Linux

On Linux the legacy process name is now set with *prctl(2)*, in addition to altering the POSIX name via `argv[0]`, as Perl has done since version 4.000. Now system utilities that read the legacy process name such as *ps*, *top*, and *killall* recognize the name you set when assigning to `$0`. The string you supply is truncated at 16 bytes; this limitation is imposed by Linux.

srand() now returns the seed

This allows programs that need to have repeatable results not to have to come up with their own seed-generating mechanism. Instead, they can use srand() and stash the return value for future use.

One example is a test program with too many combinations to test comprehensively in the time available for each run. It can test a random subset each time and, should there be a failure, log the seed used for that run so this can later be used to produce the same results.

printf-like functions understand post-1980 size modifiers

Perl's printf and sprintf operators, and Perl's internal printf replacement function, now understand the C90 size modifiers "hh" (`char`), "z" (`size_t`), and "t" (`ptrdiff_t`). Also, when compiled with a C99 compiler, Perl now understands the size modifier "j" (`intmax_t`) (but this is not portable).

So, for example, on any modern machine, `sprintf("%hhd", 257)` returns "1".

New global variable ${^GLOBAL_PHASE}

A new global variable, `${^GLOBAL_PHASE}`, has been added to allow introspection of the current phase of the Perl interpreter. It's explained in detail in *"${^GLOBAL_PHASE}" in perlvar* and in *"BEGIN, UNITCHECK, CHECK, INIT and END" in perlmod*.

-d:-foo calls Devel::foo::unimport

The syntax **-d:foo** was extended in 5.6.1 to make **-d:foo=bar** equivalent to **-MDevel::foo=bar**, which expands internally to `use Devel::foo 'bar'`. Perl now allows prefixing the module name with **-**, with the same semantics as **-M**; that is:

`-d:-foo`

> Equivalent to **-M-Devel::foo**: expands to `no Devel::foo` and calls `Devel::foo->unimport()` if that method exists.

`-d:-foo=bar`

> Equivalent to **-M-Devel::foo=bar**: expands to `no Devel::foo 'bar'`, and calls `Devel::foo->unimport("bar")` if that method exists.

This is particularly useful for suppressing the default actions of a `Devel::*` module's `import` method whilst still loading it for debugging.

Filehandle method calls load IO::File on demand

When a method call on a filehandle would die because the method cannot be resolved and *IO::File* has not been loaded, Perl now loads *IO::File* via `require` and attempts method resolution again:

```
open my $fh, ">", $file;
$fh->binmode(":raw");     # loads IO::File and succeeds
```

This also works for globs like `STDOUT`, `STDERR`, and `STDIN`:

```
STDOUT->autoflush(1);
```

Because this on-demand load happens only if method resolution fails, the legacy approach of manually loading an *IO::File* parent class for partial method support still works as expected:

```
use IO::Handle;
open my $fh, ">", $file;
$fh->autoflush(1);        # IO::File not loaded
```

Improved IPv6 support

The `Socket` module provides new affordances for IPv6, including implementations of the `Socket::getaddrinfo()` and `Socket::getnameinfo()` functions, along with related constants and a handful of new functions. See *Socket*.

DTrace probes now include package name

The `DTrace` probes now include an additional argument, `arg3`, which contains the package the subroutine being entered or left was compiled in.

For example, using the following DTrace script:

```
perl$target:::sub-entry
{
    printf("%s::%s\n", copyinstr(arg0), copyinstr(arg3));
}
```

and then running:

```
$ perl -e 'sub test { }; test'
```

`DTrace` will print:

```
main::test
```

New C APIs

See *Internal Changes*.

Security

User-defined regular expression properties

"User-Defined Character Properties" in perlunicode documented that you can create custom properties by defining subroutines whose names begin with "In" or "Is". However, Perl did not actually enforce that naming restriction, so `\p{foo::bar}` could call foo::bar() if it existed. The documented convention is now enforced.

Also, Perl no longer allows tainted regular expressions to invoke a user-defined property. It simply dies instead [perl #82616].

Incompatible Changes

Perl 5.14.0 is not binary-compatible with any previous stable release.

In addition to the sections that follow, see *C API Changes*.

Regular Expressions and String Escapes

Inverted bracketed character classes and multi-character folds

Some characters match a sequence of two or three characters in `/i` regular expression matching under Unicode rules. One example is `LATIN SMALL LETTER SHARP S` which matches the sequence `ss`.

```
 'ss' =~ /\A[\N{LATIN SMALL LETTER SHARP S}]\z/i  # Matches
```

This, however, can lead to very counter-intuitive results, especially when inverted. Because of this, Perl 5.14 does not use multi-character `/i` matching in inverted character classes.

```
 'ss' =~ /\A[^\N{LATIN SMALL LETTER SHARP S}]+\z/i  # ???
```

This should match any sequences of characters that aren't the `SHARP S` nor what `SHARP S` matches under `/i`. `"s"` isn't `SHARP S`, but Unicode says that `"ss"` is what `SHARP S` matches under `/i`. So which one "wins"? Do you fail the match because the string has `ss` or accept it because it has an `s` followed by another `s`?

Earlier releases of Perl did allow this multi-character matching, but due to bugs, it mostly did not work.

\400-\777

In certain circumstances, `\400-\777` in regexes have behaved differently than they behave in all other doublequote-like contexts. Since 5.10.1, Perl has issued a deprecation warning when this happens. Now, these literals behave the same in all doublequote-like contexts, namely to be equivalent to `\x{100}-\x{1FF}`, with no deprecation warning.

Use of `\400-\777` in the command-line option **-0** retain their conventional meaning. They slurp whole input files; previously, this was documented only for **-0777**.

Because of various ambiguities, you should use the new `\o{...}` construct to represent characters in octal instead.

Most \p{} properties are now immune to case-insensitive matching

For most Unicode properties, it doesn't make sense to have them match differently under `/i` case-insensitive matching. Doing so can lead to unexpected results and potential security holes. For example

```
m/\p{ASCII_Hex_Digit}+/i
```

could previously match non-ASCII characters because of the Unicode matching rules (although there were several bugs with this). Now matching under `/i` gives the same results as non-`/i` matching except for those few properties where people have come to expect differences, namely the ones where casing is an integral part of their meaning, such as `m/\p{Uppercase}/i` and `m/\p{Lowercase}/i`, both of which match the same code points as matched by `m/\p{Cased}/i`. Details are in *"Unicode Properties" in perlrecharclass*.

User-defined property handlers that need to match differently under `/i` must be changed to read the new boolean parameter passed to them, which is non-zero if case-insensitive matching is in effect and 0 otherwise. See *"User-Defined Character Properties" in perlunicode*.

\p{} implies Unicode semantics

Specifying a Unicode property in the pattern indicates that the pattern is meant for matching according to Unicode rules, the way `\N{`*NAME*`}` does.

Regular expressions retain their localeness when interpolated

Regular expressions compiled under `use locale` now retain this when interpolated into a new regular expression compiled outside a `use locale`, and vice-versa.

Previously, one regular expression interpolated into another inherited the localeness of the surrounding regex, losing whatever state it originally had. This is considered a bug fix, but may trip up code that has come to rely on the incorrect behaviour.

Stringification of regexes has changed

Default regular expression modifiers are now notated using `(?^...)`. Code relying on the old stringification will fail. This is so that when new modifiers are added, such code won't have to keep changing each time this happens, because the stringification will automatically incorporate the new modifiers.

Code that needs to work properly with both old- and new-style regexes can avoid the whole issue by using (for perls since 5.9.5; see *re*):

```
use re qw(regexp_pattern);
my ($pat, $mods) = regexp_pattern($re_ref);
```

If the actual stringification is important or older Perls need to be supported, you can use something like the following:

```
    # Accept both old and new-style stringification
```

```
my $modifiers = (qr/foobar/ =~ /\Q(?^/) ? "^" : "-xism";
```

And then use `$modifiers` instead of `-xism`.

Run-time code blocks in regular expressions inherit pragmata

Code blocks in regular expressions (`(?{...})` and `(??{...})`) previously did not inherit pragmata (strict, warnings, etc.) if the regular expression was compiled at run time as happens in cases like these two:

```
use re "eval";
$foo =~ $bar; # when $bar contains (?{...})
$foo =~ /$bar(?{ $finished = 1 })/;
```

This bug has now been fixed, but code that relied on the buggy behaviour may need to be fixed to account for the correct behaviour.

Stashes and Package Variables

Localised tied hashes and arrays are no longed tied

In the following:

```
tie @a, ...;
{
 local @a;
 # here, @a is a now a new, untied array
}
# here, @a refers again to the old, tied array
```

Earlier versions of Perl incorrectly tied the new local array. This has now been fixed. This fix could however potentially cause a change in behaviour of some code.

Stashes are now always defined

`defined %Foo::` now always returns true, even when no symbols have yet been defined in that package.

This is a side-effect of removing a special-case kludge in the tokeniser, added for 5.10.0, to hide side-effects of changes to the internal storage of hashes. The fix drastically reduces hashes' memory overhead.

Calling defined on a stash has been deprecated since 5.6.0, warned on lexicals since 5.6.0, and warned for stashes and other package variables since 5.12.0. `defined %hash` has always exposed an implementation detail: emptying a hash by deleting all entries from it does not make `defined %hash` false. Hence `defined %hash` is not valid code to determine whether an arbitrary hash is empty. Instead, use the behaviour of an empty `%hash` always returning false in scalar context.

Clearing stashes

Stash list assignment `%foo:: = ()` used to make the stash temporarily anonymous while it was being emptied. Consequently, any of its subroutines referenced elsewhere would become anonymous, showing up as "(unknown)" in `caller`. They now retain their package names such that `caller` returns the original sub name if there is still a reference to its typeglob and "foo::__ANON__" otherwise [perl #79208].

Dereferencing typeglobs

If you assign a typeglob to a scalar variable:

```
$glob = *foo;
```

the glob that is copied to `$glob` is marked with a special flag indicating that the glob is just a copy. This allows subsequent assignments to `$glob` to overwrite the glob. The original glob, however, is immutable.

Some Perl operators did not distinguish between these two types of globs. This would result in strange behaviour in edge cases: `untie $scalar` would not untie the scalar if the last thing assigned to it was a glob (because it treated it as `untie *$scalar`, which unties a handle). Assignment to a glob slot (such as `*$glob = \@some_array`) would simply assign `\@some_array` to `$glob`.

To fix this, the `*{}` operator (including its `*foo` and `*$foo` forms) has been modified to make a new immutable glob if its operand is a glob copy. This allows operators that make a distinction between globs and scalars to be modified to treat only immutable globs as globs. (`tie`, `tied` and `untie` have been left as they are for compatibility's sake, but will warn. See *Deprecations*.)

This causes an incompatible change in code that assigns a glob to the return value of `*{}` when that operator was passed a glob copy. Take the following code, for instance:

```
$glob = *foo;
*$glob = *bar;
```

The `*$glob` on the second line returns a new immutable glob. That new glob is made an alias to `*bar`. Then it is discarded. So the second assignment has no effect.

See *http://rt.perl.org/rt3/Public/Bug/Display.html?id=77810* for more detail.

Magic variables outside the main package

In previous versions of Perl, magic variables like `$!`, `%SIG`, etc. would "leak" into other packages. So `%foo::SIG` could be used to access signals, `${"foo::!"}` (with strict mode off) to access C's `errno`, etc.

This was a bug, or an "unintentional" feature, which caused various ill effects, such as signal handlers being wiped when modules were loaded, etc.

This has been fixed (or the feature has been removed, depending on how you see it).

local($_) strips all magic from $_

local() on scalar variables gives them a new value but keeps all their magic intact. This has proven problematic for the default scalar variable $_, where *perlsub* recommends that any subroutine that assigns to $_ should first localize it. This would throw an exception if $_ is aliased to a read-only variable, and could in general have various unintentional side-effects.

Therefore, as an exception to the general rule, local($_) will not only assign a new value to $_, but also remove all existing magic from it as well.

Parsing of package and variable names

Parsing the names of packages and package variables has changed: multiple adjacent pairs of colons, as in `foo:::bar`, are now all treated as package separators.

Regardless of this change, the exact parsing of package separators has never been guaranteed and is subject to change in future Perl versions.

Changes to Syntax or to Perl Operators

given return values

`given` blocks now return the last evaluated expression, or an empty list if the block was exited by `break`. Thus you can now write:

```
my $type = do {
 given ($num) {
```

```
    break     when undef;
    "integer" when /^[+-]?[0-9]+$/;
    "float"   when /^[+-]?[0-9]+(?:\.[0-9]+)?$/;
    "unknown";
  }
};
```

See *"Return value" in perlsyn* for details.

Change in parsing of certain prototypes

Functions declared with the following prototypes now behave correctly as unary functions:

```
*
\$ \% \@ \* \&
\[...]
;$ ;*
;\$ ;\% etc.
;\[...]
```

Due to this bug fix [perl #75904], functions using the `(*)`, `(;$)` and `(;*)` prototypes are parsed with higher precedence than before. So in the following example:

```
sub foo(;$);
foo $a < $b;
```

the second line is now parsed correctly as `foo($a) < $b`, rather than `foo($a < $b)`. This happens when one of these operators is used in an unparenthesised argument:

```
< > <= >= lt gt le ge
== != <=> eq ne cmp ~~
&
| ^
&&
|| //
.. ...
?:
= += -= *= etc.
, =>
```

Smart-matching against array slices

Previously, the following code resulted in a successful match:

```
my @a = qw(a y0 z);
my @b = qw(a x0 z);
@a[0 .. $#b] ~~ @b;
```

This odd behaviour has now been fixed [perl #77468].

Negation treats strings differently from before

The unary negation operator, `-`, now treats strings that look like numbers as numbers [perl #57706].

Negative zero

Negative zero (-0.0), when converted to a string, now becomes "0" on all platforms. It used to become "-0" on some, but "0" on others.

If you still need to determine whether a zero is negative, use `sprintf("%g", $zero) =~ /^-/` or

the *Data::Float* module on CPAN.

:= is now a syntax error

Previously `my $pi := 4` was exactly equivalent to `my $pi : = 4`, with the `:` being treated as the start of an attribute list, ending before the =. The use of `:=` to mean `: =` was deprecated in 5.12.0, and is now a syntax error. This allows future use of `:=` as a new token.

Outside the core's tests for it, we find no Perl 5 code on CPAN using this construction, so we believe that this change will have little impact on real-world codebases.

If it is absolutely necessary to have empty attribute lists (for example, because of a code generator), simply avoid the error by adding a space before the =.

Change in the parsing of identifiers

Characters outside the Unicode "XIDStart" set are no longer allowed at the beginning of an identifier. This means that certain accents and marks that normally follow an alphabetic character may no longer be the first character of an identifier.

Threads and Processes

Directory handles not copied to threads

On systems other than Windows that do not have a `fchdir` function, newly-created threads no longer inherit directory handles from their parent threads. Such programs would usually have crashed anyway [perl #75154].

close on shared pipes

To avoid deadlocks, the `close` function no longer waits for the child process to exit if the underlying file descriptor is still in use by another thread. It returns true in such cases.

fork() emulation will not wait for signalled children

On Windows parent processes would not terminate until all forked children had terminated first. However, `kill("KILL", ...)` is inherently unstable on pseudo-processes, and `kill("TERM", ...)` might not get delivered if the child is blocked in a system call.

To avoid the deadlock and still provide a safe mechanism to terminate the hosting process, Perl now no longer waits for children that have been sent a SIGTERM signal. It is up to the parent process to waitpid() for these children if child-cleanup processing must be allowed to finish. However, it is also then the responsibility of the parent to avoid the deadlock by making sure the child process can't be blocked on I/O.

See *perlfork* for more information about the fork() emulation on Windows.

Configuration

Naming fixes in Policy_sh.SH may invalidate Policy.sh

Several long-standing typos and naming confusions in *Policy_sh.SH* have been fixed, standardizing on the variable names used in *config.sh*.

This will change the behaviour of *Policy.sh* if you happen to have been accidentally relying on its incorrect behaviour.

Perl source code is read in text mode on Windows

Perl scripts used to be read in binary mode on Windows for the benefit of the *ByteLoader* module (which is no longer part of core Perl). This had the side-effect of breaking various operations on the DATA filehandle, including seek()/tell(), and even simply reading from DATA after filehandles have been flushed by a call to system(), backticks, fork() etc.

The default build options for Windows have been changed to read Perl source code on Windows in text mode now. *ByteLoader* will (hopefully) be updated on CPAN to automatically handle this situation

[perl #28106].

Deprecations

See also *Deprecated C APIs*.

Omitting a space between a regular expression and subsequent word

Omitting the space between a regular expression operator or its modifiers and the following word is deprecated. For example, `m/foo/sand $bar` is for now still parsed as `m/foo/s and $bar`, but will now issue a warning.

\cX

The backslash-c construct was designed as a way of specifying non-printable characters, but there were no restrictions (on ASCII platforms) on what the character following the c could be. Now, a deprecation warning is raised if that character isn't an ASCII character. Also, a deprecation warning is raised for `"\c{"` (which is the same as simply saying `";"`).

"\b{" and "\B{"

In regular expressions, a literal `"{"` immediately following a `"\b"` (not in a bracketed character class) or a `"\B{"` is now deprecated to allow for its future use by Perl itself.

Perl 4-era .pl libraries

Perl bundles a handful of library files that predate Perl 5. This bundling is now deprecated for most of these files, which are now available from CPAN. The affected files now warn when run, if they were installed as part of the core.

This is a mandatory warning, not obeying **-X** or lexical warning bits. The warning is modelled on that supplied by *deprecate.pm* for deprecated-in-core *.pm* libraries. It points to the specific CPAN distribution that contains the *.pl* libraries. The CPAN versions, of course, do not generate the warning.

List assignment to $[

Assignment to `$[` was deprecated and started to give warnings in Perl version 5.12.0. This version of Perl (5.14) now also emits a warning when assigning to `$[` in list context. This fixes an oversight in 5.12.0.

Use of qw(...) as parentheses

Historically the parser fooled itself into thinking that `qw(...)` literals were always enclosed in parentheses, and as a result you could sometimes omit parentheses around them:

```
for $x qw(a b c) { ... }
```

The parser no longer lies to itself in this way. Wrap the list literal in parentheses like this:

```
for $x (qw(a b c)) { ... }
```

This is being deprecated because the parentheses in `for $i (1,2,3) { ... }` are not part of expression syntax. They are part of the statement syntax, with the `for` statement wanting literal parentheses. The synthetic parentheses that a qw expression acquired were only intended to be treated as part of expression syntax.

Note that this does not change the behaviour of cases like:

```
use POSIX qw(setlocale localeconv);
our @EXPORT = qw(foo bar baz);
```

where parentheses were never required around the expression.

\N{BELL}

This is because Unicode is using that name for a different character. See *Unicode Version 6.0 is now supported (mostly)* for more explanation.

?PATTERN?

?PATTERN? (without the initial `m`) has been deprecated and now produces a warning. This is to allow future use of ? in new operators. The match-once functionality is still available as `m?PATTERN?`.

Tie functions on scalars holding typeglobs

Calling a tie function (`tie`, `tied`, `untie`) with a scalar argument acts on a filehandle if the scalar happens to hold a typeglob.

This is a long-standing bug that will be removed in Perl 5.16, as there is currently no way to tie the scalar itself when it holds a typeglob, and no way to untie a scalar that has had a typeglob assigned to it.

Now there is a deprecation warning whenever a tie function is used on a handle without an explicit `*`.

User-defined case-mapping

This feature is being deprecated due to its many issues, as documented in *"User-Defined Case Mappings (for serious hackers only)" in perlunicode*. This feature will be removed in Perl 5.16. Instead use the CPAN module *Unicode::Casing*, which provides improved functionality.

Deprecated modules

The following module will be removed from the core distribution in a future release, and should be installed from CPAN instead. Distributions on CPAN that require this should add it to their prerequisites. The core version of these module now issues a deprecation warning.

If you ship a packaged version of Perl, either alone or as part of a larger system, then you should carefully consider the repercussions of core module deprecations. You may want to consider shipping your default build of Perl with a package for the deprecated module that installs into `vendor` or `site` Perl library directories. This will inhibit the deprecation warnings.

Alternatively, you may want to consider patching *lib/deprecate.pm* to provide deprecation warnings specific to your packaging system or distribution of Perl, consistent with how your packaging system or distribution manages a staged transition from a release where the installation of a single package provides the given functionality, to a later release where the system administrator needs to know to install multiple packages to get that same functionality.

You can silence these deprecation warnings by installing the module in question from CPAN. To install the latest version of it by role rather than by name, just install `Task::Deprecations::5_14`.

Devel::DProf

We strongly recommend that you install and use *Devel::NYTProf* instead of *Devel::DProf*, as *Devel::NYTProf* offers significantly improved profiling and reporting.

Performance Enhancements

"Safe signals" optimisation

Signal dispatch has been moved from the runloop into control ops. This should give a few percent speed increase, and eliminates nearly all the speed penalty caused by the introduction of "safe signals" in 5.8.0. Signals should still be dispatched within the same statement as they were previously. If this does *not* happen, or if you find it possible to create uninterruptible loops, this is a bug, and reports are encouraged of how to recreate such issues.

Optimisation of shift() and pop() calls without arguments

Two fewer OPs are used for shift() and pop() calls with no argument (with implicit `@_`). This change makes shift() 5% faster than `shift @_` on non-threaded perls, and 25% faster on threaded ones.

Optimisation of regexp engine string comparison work

The `foldEQ_utf8` API function for case-insensitive comparison of strings (which is used heavily by the regexp engine) was substantially refactored and optimised -- and its documentation much improved as a free bonus.

Regular expression compilation speed-up

Compiling regular expressions has been made faster when upgrading the regex to utf8 is necessary but this isn't known when the compilation begins.

String appending is 100 times faster

When doing a lot of string appending, perls built to use the system's `malloc` could end up allocating a lot more memory than needed in a inefficient way.

`sv_grow`, the function used to allocate more memory if necessary when appending to a string, has been taught to round up the memory it requests to a certain geometric progression, making it much faster on certain platforms and configurations. On Win32, it's now about 100 times faster.

Eliminate PL_* accessor functions under ithreads

When `MULTIPLICITY` was first developed, and interpreter state moved into an interpreter struct, thread- and interpreter-local `PL_*` variables were defined as macros that called accessor functions (returning the address of the value) outside the Perl core. The intent was to allow members within the interpreter struct to change size without breaking binary compatibility, so that bug fixes could be merged to a maintenance branch that necessitated such a size change. This mechanism was redundant and penalised well-behaved code. It has been removed.

Freeing weak references

When there are many weak references to an object, freeing that object can under some circumstances take O(*N***N*) time to free, where *N* is the number of references. The circumstances in which this can happen have been reduced [perl #75254]

Lexical array and hash assignments

An earlier optimisation to speed up `my @array = ...` and `my %hash = ...` assignments caused a bug and was disabled in Perl 5.12.0.

Now we have found another way to speed up these assignments [perl #82110].

@_ uses less memory

Previously, `@_` was allocated for every subroutine at compile time with enough space for four entries. Now this allocation is done on demand when the subroutine is called [perl #72416].

Size optimisations to SV and HV structures

`xhv_fill` has been eliminated from `struct xpvhv`, saving 1 IV per hash and on some systems will cause `struct xpvhv` to become cache-aligned. To avoid this memory saving causing a slowdown elsewhere, boolean use of `HvFILL` now calls `HvTOTALKEYS` instead (which is equivalent), so while the fill data when actually required are now calculated on demand, cases when this needs to be done should be rare.

The order of structure elements in SV bodies has changed. Effectively, the NV slot has swapped location with STASH and MAGIC. As all access to SV members is via macros, this should be completely transparent. This change allows the space saving for PVHVs documented above, and may reduce the memory allocation needed for PVIVs on some architectures.

`XPV`, `XPVIV`, and `XPVNV` now allocate only the parts of the SV body they actually use, saving some space.

Scalars containing regular expressions now allocate only the part of the SV body they actually use, saving some space.

Memory consumption improvements to Exporter

The `@EXPORT_FAIL` AV is no longer created unless needed, hence neither is the typeglob backing it. This saves about 200 bytes for every package that uses Exporter but doesn't use this functionality.

Memory savings for weak references

For weak references, the common case of just a single weak reference per referent has been optimised to reduce the storage required. In this case it saves the equivalent of one small Perl array per referent.

%+ and %- use less memory

The bulk of the `Tie::Hash::NamedCapture` module used to be in the Perl core. It has now been moved to an XS module to reduce overhead for programs that do not use `%+` or `%-`.

Multiple small improvements to threads

The internal structures of threading now make fewer API calls and fewer allocations, resulting in noticeably smaller object code. Additionally, many thread context checks have been deferred so they're done only as needed (although this is only possible for non-debugging builds).

Adjacent pairs of nextstate opcodes are now optimized away

Previously, in code such as

```
use constant DEBUG => 0;

sub GAK {
    warn if DEBUG;
    print "stuff\n";
}
```

the ops for `warn if DEBUG` would be folded to a `null` op (`ex-const`), but the `nextstate` op would remain, resulting in a runtime op dispatch of `nextstate`, `nextstate`, etc.

The execution of a sequence of `nextstate` ops is indistinguishable from just the last `nextstate` op so the peephole optimizer now eliminates the first of a pair of `nextstate` ops except when the first carries a label, since labels must not be eliminated by the optimizer, and label usage isn't conclusively known at compile time.

Modules and Pragmata

New Modules and Pragmata

- *CPAN::Meta::YAML* 0.003 has been added as a dual-life module. It supports a subset of YAML sufficient for reading and writing *META.yml* and *MYMETA.yml* files included with CPAN distributions or generated by the module installation toolchain. It should not be used for any other general YAML parsing or generation task.
- *CPAN::Meta* version 2.110440 has been added as a dual-life module. It provides a standard library to read, interpret and write CPAN distribution metadata files (like *META.json* and *META.yml)* that describe a distribution, its contents, and the requirements for building it and installing it. The latest CPAN distribution metadata specification is included as *CPAN::Meta::Spec* and notes on changes in the specification over time are given in *CPAN::Meta::History*.
- *HTTP::Tiny* 0.012 has been added as a dual-life module. It is a very small, simple HTTP/1.1 client designed for simple GET requests and file mirroring. It has been added so that *CPAN.pm* and *CPANPLUS* can "bootstrap" HTTP access to CPAN using pure Perl without relying on external binaries like *curl(1)* or *wget(1)*.
- *JSON::PP* 2.27105 has been added as a dual-life module to allow CPAN clients to read

META.json files in CPAN distributions.

- *Module::Metadata* 1.000004 has been added as a dual-life module. It gathers package and POD information from Perl module files. It is a standalone module based on *Module::Build::ModuleInfo* for use by other module installation toolchain components. *Module::Build::ModuleInfo* has been deprecated in favor of this module instead.
- *Perl::OSType* 1.002 has been added as a dual-life module. It maps Perl operating system names (like "dragonfly" or "MSWin32") to more generic types with standardized names (like "Unix" or "Windows"). It has been refactored out of *Module::Build* and *ExtUtils::CBuilder* and consolidates such mappings into a single location for easier maintenance.
- The following modules were added by the *Unicode::Collate* upgrade. See below for details.

  Unicode::Collate::CJK::Big5

  Unicode::Collate::CJK::GB2312

  Unicode::Collate::CJK::JISX0208

  Unicode::Collate::CJK::Korean

  Unicode::Collate::CJK::Pinyin

  Unicode::Collate::CJK::Stroke
- *Version::Requirements* version 0.101020 has been added as a dual-life module. It provides a standard library to model and manipulates module prerequisites and version constraints defined in *CPAN::Meta::Spec*.

Updated Modules and Pragma

- *attributes* has been upgraded from version 0.12 to 0.14.
- *Archive::Extract* has been upgraded from version 0.38 to 0.48.

  Updates since 0.38 include: a safe print method that guards *Archive::Extract* from changes to $\; a fix to the tests when run in core Perl; support for TZ files; a modification for the lzma logic to favour *IO::Uncompress::Unlzma*; and a fix for an issue with NetBSD-current and its new *unzip(1)* executable.
- *Archive::Tar* has been upgraded from version 1.54 to 1.76.

  Important changes since 1.54 include the following:

  - Compatibility with busybox implementations of *tar(1)*.
  - A fix so that write() and create_archive() close only filehandles they themselves opened.
  - A bug was fixed regarding the exit code of extract_archive.
  - The *ptar(1)* utility has a new option to allow safe creation of tarballs without world-writable files on Windows, allowing those archives to be uploaded to CPAN.
  - A new *ptargrep(1)* utility for using regular expressions against the contents of files in a tar archive.
  - *pax* extended headers are now skipped.
- *Attribute::Handlers* has been upgraded from version 0.87 to 0.89.
- *autodie* has been upgraded from version 2.06_01 to 2.1001.
- *AutoLoader* has been upgraded from version 5.70 to 5.71.
- The *B* module has been upgraded from version 1.23 to 1.29.

It no longer crashes when taking apart a `y///` containing characters outside the octet range or compiled in a `use utf8` scope.

The size of the shared object has been reduced by about 40%, with no reduction in functionality.

- *B::Concise* has been upgraded from version 0.78 to 0.83.

  B::Concise marks rv2sv(), rv2av(), and rv2hv() ops with the new `OPpDEREF` flag as "DREFed".

  It no longer produces mangled output with the **-tree** option [perl #80632].

- *B::Debug* has been upgraded from version 1.12 to 1.16.

- *B::Deparse* has been upgraded from version 0.96 to 1.03.

  The deparsing of a `nextstate` op has changed when it has both a change of package relative to the previous nextstate, or a change of `%^H` or other state and a label. The label was previously emitted first, but is now emitted last (5.12.1).

  The `no 5.13.2` or similar form is now correctly handled by *B::Deparse* (5.12.3).

  B::Deparse now properly handles the code that applies a conditional pattern match against implicit $_ as it was fixed in [perl #20444].

  Deparsing of `our` followed by a variable with funny characters (as permitted under the `use utf8` pragma) has also been fixed [perl #33752].

- *B::Lint* has been upgraded from version 1.11_01 to 1.13.

- *base* has been upgraded from version 2.15 to 2.16.

- *Benchmark* has been upgraded from version 1.11 to 1.12.

- *bignum* has been upgraded from version 0.23 to 0.27.

- *Carp* has been upgraded from version 1.15 to 1.20.

  Carp now detects incomplete *caller()* overrides and avoids using bogus `@DB::args`. To provide backtraces, Carp relies on particular behaviour of the caller() builtin. *Carp* now detects if other code has overridden this with an incomplete implementation, and modifies its backtrace accordingly. Previously incomplete overrides would cause incorrect values in backtraces (best case), or obscure fatal errors (worst case).

  This fixes certain cases of "Bizarre copy of ARRAY" caused by modules overriding caller() incorrectly (5.12.2).

  It now also avoids using regular expressions that cause Perl to load its Unicode tables, so as to avoid the "BEGIN not safe after errors" error that ensue if there has been a syntax error [perl #82854].

- *CGI* has been upgraded from version 3.48 to 3.52.

  This provides the following security fixes: the MIME boundary in multipart_init() is now random and the handling of newlines embedded in header values has been improved.

- *Compress::Raw::Bzip2* has been upgraded from version 2.024 to 2.033.

  It has been updated to use *bzip2(1)* 1.0.6.

- *Compress::Raw::Zlib* has been upgraded from version 2.024 to 2.033.

- *constant* has been upgraded from version 1.20 to 1.21.

  Unicode constants work once more. They have been broken since Perl 5.10.0 [CPAN RT #67525].

- *CPAN* has been upgraded from version 1.94_56 to 1.9600.

  Major highlights:

* much less configuration dialog hassle

* support for *META/MYMETA.json*

* support for *local::lib*

* support for *HTTP::Tiny* to reduce the dependency on FTP sites

* automatic mirror selection

* iron out all known bugs in configure_requires

* support for distributions compressed with *bzip2(1)*

* allow *Foo/Bar.pm* on the command line to mean `Foo::Bar`

- *CPANPLUS* has been upgraded from version 0.90 to 0.9103.

  A change to *cpanp-run-perl* resolves *RT #55964* and *RT #57106*, both of which related to failures to install distributions that use `Module::Install::DSL` (5.12.2).

  A dependency on *Config* was not recognised as a core module dependency. This has been fixed.

  CPANPLUS now includes support for *META.json* and *MYMETA.json*.

- *CPANPLUS::Dist::Build* has been upgraded from version 0.46 to 0.54.

- *Data::Dumper* has been upgraded from version 2.125 to 2.130_02.

  The indentation used to be off when `$Data::Dumper::Terse` was set. This has been fixed [perl #73604].

  This upgrade also fixes a crash when using custom sort functions that might cause the stack to change [perl #74170].

  Dumpxs no longer crashes with globs returned by `*$io_ref` [perl #72332].

- *DB_File* has been upgraded from version 1.820 to 1.821.

- *DBM_Filter* has been upgraded from version 0.03 to 0.04.

- *Devel::DProf* has been upgraded from version 20080331.00 to 20110228.00.

  Merely loading *Devel::DProf* now no longer triggers profiling to start. Both `use Devel::DProf` and `perl -d:DProf ...` behave as before and start the profiler.

  NOTE: *Devel::DProf* is deprecated and will be removed from a future version of Perl. We strongly recommend that you install and use *Devel::NYTProf* instead, as it offers significantly improved profiling and reporting.

- *Devel::Peek* has been upgraded from version 1.04 to 1.07.

- *Devel::SelfStubber* has been upgraded from version 1.03 to 1.05.

- *diagnostics* has been upgraded from version 1.19 to 1.22.

  It now renders pod links slightly better, and has been taught to find descriptions for messages that share their descriptions with other messages.

- *Digest::MD5* has been upgraded from version 2.39 to 2.51.

  It is now safe to use this module in combination with threads.

- *Digest::SHA* has been upgraded from version 5.47 to 5.61.

  `shasum` now more closely mimics *sha1sum(1)*/*md5sum(1)*.

  `addfile` accepts all POSIX filenames.

  New SHA-512/224 and SHA-512/256 transforms (ref. NIST Draft FIPS 180-4 [February 2011])

- *DirHandle* has been upgraded from version 1.03 to 1.04.

- *Dumpvalue* has been upgraded from version 1.13 to 1.16.
- *DynaLoader* has been upgraded from version 1.10 to 1.13.

  It fixes a buffer overflow when passed a very long file name.

  It no longer inherits from *AutoLoader*; hence it no longer produces weird error messages for unsuccessful method calls on classes that inherit from *DynaLoader* [perl #84358].
- *Encode* has been upgraded from version 2.39 to 2.42.

  Now, all 66 Unicode non-characters are treated the same way U+FFFF has always been treated: in cases when it was disallowed, all 66 are disallowed, and in cases where it warned, all 66 warn.
- *Env* has been upgraded from version 1.01 to 1.02.
- *Errno* has been upgraded from version 1.11 to 1.13.

  The implementation of *Errno* has been refactored to use about 55% less memory.

  On some platforms with unusual header files, like Win32 *gcc(1)* using `mingw64` headers, some constants that weren't actually error numbers have been exposed by *Errno*. This has been fixed [perl #77416].
- *Exporter* has been upgraded from version 5.64_01 to 5.64_03.

  Exporter no longer overrides `$SIG{__WARN__}` [perl #74472]
- *ExtUtils::CBuilder* has been upgraded from version 0.27 to 0.280203.
- *ExtUtils::Command* has been upgraded from version 1.16 to 1.17.
- *ExtUtils::Constant* has been upgraded from 0.22 to 0.23.

  The *AUTOLOAD* helper code generated by `ExtUtils::Constant::ProxySubs` can now croak() for missing constants, or generate a complete `AUTOLOAD` subroutine in XS, allowing simplification of many modules that use it (*Fcntl*, *File::Glob*, *GDBM_File*, *I18N::Langinfo*, *POSIX*, *Socket*).

  ExtUtils::Constant::ProxySubs can now optionally push the names of all constants onto the package's `@EXPORT_OK`.
- *ExtUtils::Install* has been upgraded from version 1.55 to 1.56.
- *ExtUtils::MakeMaker* has been upgraded from version 6.56 to 6.57_05.
- *ExtUtils::Manifest* has been upgraded from version 1.57 to 1.58.
- *ExtUtils::ParseXS* has been upgraded from version 2.21 to 2.2210.
- *Fcntl* has been upgraded from version 1.06 to 1.11.
- *File::Basename* has been upgraded from version 2.78 to 2.82.
- *File::CheckTree* has been upgraded from version 4.4 to 4.41.
- *File::Copy* has been upgraded from version 2.17 to 2.21.
- *File::DosGlob* has been upgraded from version 1.01 to 1.04.

  It allows patterns containing literal parentheses: they no longer need to be escaped. On Windows, it no longer adds an extra *./* to file names returned when the pattern is a relative glob with a drive specification, like *C:*.pl* [perl #71712].
- *File::Fetch* has been upgraded from version 0.24 to 0.32.

  HTTP::Lite is now supported for the "http" scheme.

The *fetch(1)* utility is supported on FreeBSD, NetBSD, and Dragonfly BSD for the `http` and `ftp` schemes.

- *File::Find* has been upgraded from version 1.15 to 1.19.

  It improves handling of backslashes on Windows, so that paths like *C:\dir\file* are no longer generated [perl #71710].

- *File::Glob* has been upgraded from version 1.07 to 1.12.

- *File::Spec* has been upgraded from version 3.31 to 3.33.

  Several portability fixes were made in *File::Spec::VMS*: a colon is now recognized as a delimiter in native filespecs; caret-escaped delimiters are recognized for better handling of extended filespecs; catpath() returns an empty directory rather than the current directory if the input directory name is empty; and abs2rel() properly handles Unix-style input (5.12.2).

- *File::stat* has been upgraded from 1.02 to 1.05.

  The `-x` and `-X` file test operators now work correctly when run by the superuser.

- *Filter::Simple* has been upgraded from version 0.84 to 0.86.

- *GDBM_File* has been upgraded from 1.10 to 1.14.

  This fixes a memory leak when DBM filters are used.

- *Hash::Util* has been upgraded from 0.07 to 0.11.

  Hash::Util no longer emits spurious "uninitialized" warnings when recursively locking hashes that have undefined values [perl #74280].

- *Hash::Util::FieldHash* has been upgraded from version 1.04 to 1.09.

- *I18N::Collate* has been upgraded from version 1.01 to 1.02.

- *I18N::Langinfo* has been upgraded from version 0.03 to 0.08.

  langinfo() now defaults to using $_ if there is no argument given, just as the documentation has always claimed.

- *I18N::LangTags* has been upgraded from version 0.35 to 0.35_01.

- *if* has been upgraded from version 0.05 to 0.0601.

- *IO* has been upgraded from version 1.25_02 to 1.25_04.

  This version of *IO* includes a new *IO::Select*, which now allows *IO::Handle* objects (and objects in derived classes) to be removed from an *IO::Select* set even if the underlying file descriptor is closed or invalid.

- *IPC::Cmd* has been upgraded from version 0.54 to 0.70.

  Resolves an issue with splitting Win32 command lines. An argument consisting of the single character "0" used to be omitted (CPAN RT #62961).

- *IPC::Open3* has been upgraded from 1.05 to 1.09.

  open3() now produces an error if the `exec` call fails, allowing this condition to be distinguished from a child process that exited with a non-zero status [perl #72016].

  The internal xclose() routine now knows how to handle file descriptors as documented, so duplicating `STDIN` in a child process using its file descriptor now works [perl #76474].

- *IPC::SysV* has been upgraded from version 2.01 to 2.03.

- *lib* has been upgraded from version 0.62 to 0.63.

- *Locale::Maketext* has been upgraded from version 1.14 to 1.19.

Locale::Maketext now supports external caches.

This upgrade also fixes an infinite loop in `Locale::Maketext::Guts::_compile()` when working with tainted values (CPAN RT #40727).

`->maketext` calls now back up and restore `$@` so error messages are not suppressed (CPAN RT #34182).

- *Log::Message* has been upgraded from version 0.02 to 0.04.
- *Log::Message::Simple* has been upgraded from version 0.06 to 0.08.
- *Math::BigInt* has been upgraded from version 1.89_01 to 1.994.

  This fixes, among other things, incorrect results when computing binomial coefficients [perl #77640].

  It also prevents `sqrt($int)` from crashing under `use bigrat`. [perl #73534].
- *Math::BigInt::FastCalc* has been upgraded from version 0.19 to 0.28.
- *Math::BigRat* has been upgraded from version 0.24 to 0.26_02.
- *Memoize* has been upgraded from version 1.01_03 to 1.02.
- *MIME::Base64* has been upgraded from 3.08 to 3.13.

  Includes new functions to calculate the length of encoded and decoded base64 strings.

  Now provides encode_base64url() and decode_base64url() functions to process the base64 scheme for "URL applications".
- *Module::Build* has been upgraded from version 0.3603 to 0.3800.

  A notable change is the deprecation of several modules. *Module::Build::Version* has been deprecated and *Module::Build* now relies on the *version* pragma directly. *Module::Build::ModuleInfo* has been deprecated in favor of a standalone copy called *Module::Metadata*. *Module::Build::YAML* has been deprecated in favor of *CPAN::Meta::YAML*.

  Module::Build now also generates *META.json* and *MYMETA.json* files in accordance with version 2 of the CPAN distribution metadata specification, *CPAN::Meta::Spec*. The older format *META.yml* and *MYMETA.yml* files are still generated.
- *Module::CoreList* has been upgraded from version 2.29 to 2.47.

  Besides listing the updated core modules of this release, it also stops listing the `Filespec` module. That module never existed in core. The scripts generating *Module::CoreList* confused it with *VMS::Filespec*, which actually is a core module as of Perl 5.8.7.
- *Module::Load* has been upgraded from version 0.16 to 0.18.
- *Module::Load::Conditional* has been upgraded from version 0.34 to 0.44.
- The *mro* pragma has been upgraded from version 1.02 to 1.07.
- *NDBM_File* has been upgraded from version 1.08 to 1.12.

  This fixes a memory leak when DBM filters are used.
- *Net::Ping* has been upgraded from version 2.36 to 2.38.
- *NEXT* has been upgraded from version 0.64 to 0.65.
- *Object::Accessor* has been upgraded from version 0.36 to 0.38.
- *ODBM_File* has been upgraded from version 1.07 to 1.10.

  This fixes a memory leak when DBM filters are used.

- *Opcode* has been upgraded from version 1.15 to 1.18.
- The *overload* pragma has been upgraded from 1.10 to 1.13.

  `overload::Method` can now handle subroutines that are themselves blessed into overloaded classes [perl #71998].

  The documentation has greatly improved. See *Documentation* below.
- *Params::Check* has been upgraded from version 0.26 to 0.28.
- The *parent* pragma has been upgraded from version 0.223 to 0.225.
- *Parse::CPAN::Meta* has been upgraded from version 1.40 to 1.4401.

  The latest Parse::CPAN::Meta can now read YAML and JSON files using *CPAN::Meta::YAML* and *JSON::PP*, which are now part of the Perl core.
- *PerlIO::encoding* has been upgraded from version 0.12 to 0.14.
- *PerlIO::scalar* has been upgraded from 0.07 to 0.11.

  A read() after a seek() beyond the end of the string no longer thinks it has data to read [perl #78716].
- *PerlIO::via* has been upgraded from version 0.09 to 0.11.
- *Pod::Html* has been upgraded from version 1.09 to 1.11.
- *Pod::LaTeX* has been upgraded from version 0.58 to 0.59.
- *Pod::Perldoc* has been upgraded from version 3.15_02 to 3.15_03.
- *Pod::Simple* has been upgraded from version 3.13 to 3.16.
- *POSIX* has been upgraded from 1.19 to 1.24.

  It now includes constants for POSIX signal constants.
- The *re* pragma has been upgraded from version 0.11 to 0.18.

  The `use re '/flags'` subpragma is new.

  The regmust() function used to crash when called on a regular expression belonging to a pluggable engine. Now it croaks instead.

  regmust() no longer leaks memory.
- *Safe* has been upgraded from version 2.25 to 2.29.

  Coderefs returned by reval() and rdo() are now wrapped via wrap_code_refs() (5.12.1).

  This fixes a possible infinite loop when looking for coderefs.

  It adds several `version::vxs::*` routines to the default share.
- *SDBM_File* has been upgraded from version 1.06 to 1.09.
- *SelfLoader* has been upgraded from 1.17 to 1.18.

  It now works in taint mode [perl #72062].
- The *sigtrap* pragma has been upgraded from version 1.04 to 1.05.

  It no longer tries to modify read-only arguments when generating a backtrace [perl #72340].
- *Socket* has been upgraded from version 1.87 to 1.94.

  See *Improved IPv6 support* above.
- *Storable* has been upgraded from version 2.22 to 2.27.

Includes performance improvement for overloaded classes.

This adds support for serialising code references that contain UTF-8 strings correctly. The *Storable* minor version number changed as a result, meaning that *Storable* users who set `$Storable::accept_future_minor` to a FALSE value will see errors (see *"FORWARD COMPATIBILITY" in Storable* for more details).

Freezing no longer gets confused if the Perl stack gets reallocated during freezing [perl #80074].

- *Sys::Hostname* has been upgraded from version 1.11 to 1.16.
- *Term::ANSIColor* has been upgraded from version 2.02 to 3.00.
- *Term::UI* has been upgraded from version 0.20 to 0.26.
- *Test::Harness* has been upgraded from version 3.17 to 3.23.
- *Test::Simple* has been upgraded from version 0.94 to 0.98.

  Among many other things, subtests without a `plan` or `no_plan` now have an implicit done_testing() added to them.
- *Thread::Semaphore* has been upgraded from version 2.09 to 2.12.

  It provides two new methods that give more control over the decrementing of semaphores: `down_nb` and `down_force`.
- *Thread::Queue* has been upgraded from version 2.11 to 2.12.
- The *threads* pragma has been upgraded from version 1.75 to 1.83.
- The *threads::shared* pragma has been upgraded from version 1.32 to 1.37.
- *Tie::Hash* has been upgraded from version 1.03 to 1.04.

  Calling `Tie::Hash->TIEHASH()` used to loop forever. Now it `croaks`.
- *Tie::Hash::NamedCapture* has been upgraded from version 0.06 to 0.08.
- *Tie::RefHash* has been upgraded from version 1.38 to 1.39.
- *Time::HiRes* has been upgraded from version 1.9719 to 1.9721_01.
- *Time::Local* has been upgraded from version 1.1901_01 to 1.2000.
- *Time::Piece* has been upgraded from version 1.15_01 to 1.20_01.
- *Unicode::Collate* has been upgraded from version 0.52_01 to 0.73.

  Unicode::Collate has been updated to use Unicode 6.0.0.

  Unicode::Collate::Locale now supports a plethora of new locales: *ar, be, bg, de__phonebook, hu, hy, kk, mk, nso, om, tn, vi, hr, ig, ja, ko, ru, sq, se, sr, to, uk, zh, zh__big5han, zh__gb2312han, zh__pinyin*, and *zh__stroke*.

  The following modules have been added:

  Unicode::Collate::CJK::Big5 for `zh__big5han` which makes tailoring of CJK Unified Ideographs in the order of CLDR's big5han ordering.

  Unicode::Collate::CJK::GB2312 for `zh__gb2312han` which makes tailoring of CJK Unified Ideographs in the order of CLDR's gb2312han ordering.

  Unicode::Collate::CJK::JISX0208 which makes tailoring of 6355 kanji (CJK Unified Ideographs) in the JIS X 0208 order.

  Unicode::Collate::CJK::Korean which makes tailoring of CJK Unified Ideographs in the order of CLDR's Korean ordering.

Unicode::Collate::CJK::Pinyin for `zh__pinyin` which makes tailoring of CJK Unified Ideographs in the order of CLDR's pinyin ordering.

Unicode::Collate::CJK::Stroke for `zh__stroke` which makes tailoring of CJK Unified Ideographs in the order of CLDR's stroke ordering.

This also sees the switch from using the pure-Perl version of this module to the XS version.

- *Unicode::Normalize* has been upgraded from version 1.03 to 1.10.
- *Unicode::UCD* has been upgraded from version 0.27 to 0.32.

  A new function, Unicode::UCD::num(), has been added. This function returns the numeric value of the string passed it or `undef` if the string in its entirety has no "safe" numeric value. (For more detail, and for the definition of "safe", see *"num()" in Unicode::UCD*.)

  This upgrade also includes several bug fixes:

  charinfo()

  - It is now updated to Unicode Version 6.0.0 with *Corrigendum #8*, excepting that, just as with Perl 5.14, the code point at U+1F514 has no name.
  - Hangul syllable code points have the correct names, and their decompositions are always output without requiring *Lingua::KO::Hangul::Util* to be installed.
  - CJK (Chinese-Japanese-Korean) code points U+2A700 to U+2B734 and U+2B740 to U+2B81D are now properly handled.
  - Numeric values are now output for those CJK code points that have them.
  - Names output for code points with multiple aliases are now the corrected ones.

  charscript()

  This now correctly returns "Unknown" instead of `undef` for the script of a code point that hasn't been assigned another one.

  charblock()

  This now correctly returns "No_Block" instead of `undef` for the block of a code point that hasn't been assigned to another one.

- The *version* pragma has been upgraded from 0.82 to 0.88.

  Because of a bug, now fixed, the is_strict() and is_lax() functions did not work when exported (5.12.1).

- The *warnings* pragma has been upgraded from version 1.09 to 1.12.

  Calling `use warnings` without arguments is now significantly more efficient.

- The *warnings::register* pragma has been upgraded from version 1.01 to 1.02.

  It is now possible to register warning categories other than the names of packages using *warnings::register*. See *perllexwarn(1)* for more information.

- *XSLoader* has been upgraded from version 0.10 to 0.13.
- *VMS::DCLsym* has been upgraded from version 1.03 to 1.05.

  Two bugs have been fixed [perl #84086]:

  The symbol table name was lost when tying a hash, due to a thinko in `TIEHASH`. The result was that all tied hashes interacted with the local symbol table.

  Unless a symbol table name had been explicitly specified in the call to the constructor, querying the special key `:LOCAL` failed to identify objects connected to the local symbol table.

- The *Win32* module has been upgraded from version 0.39 to 0.44.

This release has several new functions: Win32::GetSystemMetrics(), Win32::GetProductInfo(), Win32::GetOSDisplayName().

The names returned by Win32::GetOSName() and Win32::GetOSDisplayName() have been corrected.

- *XS::Typemap* has been upgraded from version 0.03 to 0.05.

Removed Modules and Pragmata

As promised in Perl 5.12.0's release notes, the following modules have been removed from the core distribution, and if needed should be installed from CPAN instead.

- *Class::ISA* has been removed from the Perl core. Prior version was 0.36.
- *Pod::Plainer* has been removed from the Perl core. Prior version was 1.02.
- *Switch* has been removed from the Perl core. Prior version was 2.16.

The removal of *Shell* has been deferred until after 5.14, as the implementation of *Shell* shipped with 5.12.0 did not correctly issue the warning that it was to be removed from core.

Documentation

New Documentation

perlgpl

perlgpl has been updated to contain GPL version 1, as is included in the *README* distributed with Perl (5.12.1).

Perl 5.12.x delta files

The perldelta files for Perl 5.12.1 to 5.12.3 have been added from the maintenance branch: *perl5121delta*, *perl5122delta*, *perl5123delta*.

perlpodstyle

New style guide for POD documentation, split mostly from the NOTES section of the *pod2man(1)* manpage.

perlsource, perlinterp, perlhacktut, and perlhacktips

See *perlhack and perlrepository revamp*, below.

Changes to Existing Documentation

perlmodlib is now complete

The *perlmodlib* manpage that came with Perl 5.12.0 was missing several modules due to a bug in the script that generates the list. This has been fixed [perl #74332] (5.12.1).

Replace incorrect tr/// table in perlebcdic

perlebcdic contains a helpful table to use in `tr///` to convert between EBCDIC and Latin1/ASCII. The table was the inverse of the one it describes, though the code that used the table worked correctly for the specific example given.

The table has been corrected and the sample code changed to correspond.

The table has also been changed to hex from octal, and the recipes in the pod have been altered to print out leading zeros to make all values the same length.

Tricks for user-defined casing

perlunicode now contains an explanation of how to override, mangle and otherwise tweak the way Perl handles upper-, lower- and other-case conversions on Unicode data, and how to provide scoped changes to alter one's own code's behaviour without stomping on anybody else's.

INSTALL explicitly states that Perl requires a C89 compiler

This was already true, but it's now Officially Stated For The Record (5.12.2).

Explanation of \xHH and \oOOO escapes

perlop has been updated with more detailed explanation of these two character escapes.

-0NNN switch

In *perlrun*, the behaviour of the **-0NNN** switch for **-0400** or higher has been clarified (5.12.2).

Maintenance policy

perlpolicy now contains the policy on what patches are acceptable for maintenance branches (5.12.1).

Deprecation policy

perlpolicy now contains the policy on compatibility and deprecation along with definitions of terms like "deprecation" (5.12.2).

New descriptions in perldiag

The following existing diagnostics are now documented:

- *Ambiguous use of %c resolved as operator %c*
- *Ambiguous use of %c{%s} resolved to %c%s*
- *Ambiguous use of %c{%s[...]} resolved to %c%s[...]*
- *Ambiguous use of %c{%s{...}} resolved to %c%s{...}*
- *Ambiguous use of -%s resolved as -&%s()*
- *Invalid strict version format (%s)*
- *Invalid version format (%s)*
- *Invalid version object*

perlbook

perlbook has been expanded to cover many more popular books.

SvTRUE macro

The documentation for the SvTRUE macro in *perlapi* was simply wrong in stating that get-magic is not processed. It has been corrected.

op manipulation functions

Several API functions that process optrees have been newly documented.

perlvar revamp

perlvar reorders the variables and groups them by topic. Each variable introduced after Perl 5.000 notes the first version in which it is available. *perlvar* also has a new section for deprecated variables to note when they were removed.

Array and hash slices in scalar context

These are now documented in *perldata*.

use locale and formats

perlform and *perllocale* have been corrected to state that `use locale` affects formats.

overload

overload's documentation has practically undergone a rewrite. It is now much more straightforward and clear.

perlhack and perlrepository revamp

The *perlhack* document is now much shorter, and focuses on the Perl 5 development process and submitting patches to Perl. The technical content has been moved to several new documents, *perlsource*, *perlinterp*, *perlhacktut*, and *perlhacktips*. This technical content has been only lightly edited.

The perlrepository document has been renamed to *perlgit*. This new document is just a how-to on using git with the Perl source code. Any other content that used to be in perlrepository has been moved to *perlhack*.

Time::Piece examples

Examples in *perlfaq4* have been updated to show the use of *Time::Piece*.

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 Errors

Closure prototype called

This error occurs when a subroutine reference passed to an attribute handler is called, if the subroutine is a closure [perl #68560].

Insecure user-defined property %s

Perl detected tainted data when trying to compile a regular expression that contains a call to a user-defined character property function, meaning `\p{IsFoo}` or `\p{InFoo}`. See *"User-Defined Character Properties" in perlunicode* and *perlsec*.

panic: gp_free failed to free glob pointer - something is repeatedly re-creating entries

This new error is triggered if a destructor called on an object in a typeglob that is being freed creates a new typeglob entry containing an object with a destructor that creates a new entry containing an object etc.

Parsing code internal error (%s)

This new fatal error is produced when parsing code supplied by an extension violates the parser's API in a detectable way.

refcnt: fd %d%s

This new error only occurs if a internal consistency check fails when a pipe is about to be closed.

Regexp modifier "/%c" may not appear twice

The regular expression pattern has one of the mutually exclusive modifiers repeated.

Regexp modifiers "/%c" and "/%c" are mutually exclusive

The regular expression pattern has more than one of the mutually exclusive modifiers.

Using !~ with %s doesn't make sense

This error occurs when `!~` is used with `s///r` or `y///r`.

New Warnings

"\b{" is deprecated; use "\b\{" instead

"\B{" is deprecated; use "\B\{" instead

Use of an unescaped "{" immediately following a `\b` or `\B` is now deprecated in order to reserve its use for Perl itself in a future release.

Operation "%s" returns its argument for ...

Performing an operation requiring Unicode semantics (such as case-folding) on a Unicode surrogate or a non-Unicode character now triggers this warning.

Use of qw(...) as parentheses is deprecated

See *Use of qw(...) as parentheses*, above, for details.

Changes to Existing Diagnostics

- The "Variable $foo is not imported" warning that precedes a `strict 'vars'` error has now been assigned the "misc" category, so that `no warnings` will suppress it [perl #73712].
- warn() and die() now produce "Wide character" warnings when fed a character outside the byte range if STDERR is a byte-sized handle.
- The "Layer does not match this perl" error message has been replaced with these more helpful messages [perl #73754]:
  - PerlIO layer function table size (%d) does not match size expected by this perl (%d)
  - PerlIO layer instance size (%d) does not match size expected by this perl (%d)
- The "Found = in conditional" warning that is emitted when a constant is assigned to a variable in a condition is now withheld if the constant is actually a subroutine or one generated by `use constant`, since the value of the constant may not be known at the time the program is written [perl #77762].
- Previously, if none of the gethostbyaddr(), gethostbyname() and gethostent() functions were implemented on a given platform, they would all die with the message "Unsupported socket function 'gethostent' called", with analogous messages for getnet*() and getserv*(). This has been corrected.
- The warning message about unrecognized regular expression escapes passed through has been changed to include any literal "{" following the two-character escape. For example, "\q{" is now emitted instead of "\q".

Utility Changes

perlbug(1)

- *perlbug* now looks in the EMAIL environment variable for a return address if the REPLY-TO and REPLYTO variables are empty.
- *perlbug* did not previously generate a "From:" header, potentially resulting in dropped mail; it now includes that header.
- The user's address is now used as the Return-Path.

  Many systems these days don't have a valid Internet domain name, and perlbug@perl.org does not accept email with a return-path that does not resolve. So the user's address is now passed to sendmail so it's less likely to get stuck in a mail queue somewhere [perl #82996].
- *perlbug* now always gives the reporter a chance to change the email address it guesses for them (5.12.2).
- *perlbug* should no longer warn about uninitialized values when using the **-d** and **-v** options

perl5db.pl (5.12.2).

- The remote terminal works after forking and spawns new sessions, one per forked process.

ptargrep

- *ptargrep* is a new utility to apply pattern matching to the contents of files in a tar archive. It comes with `Archive::Tar`.

Configuration and Compilation

See also *Naming fixes in Policy_sh.SH may invalidate Policy.sh*, above.

- CCINCDIR and CCLIBDIR for the mingw64 cross-compiler are now correctly under *$(CCHOME)\mingw\include* and *\lib* rather than immediately below *$(CCHOME)*.

  This means the "incpath", "libpth", "ldflags", "lddlflags" and "ldflags_nolargefiles" values in *Config.pm* and *Config_heavy.pl* are now set correctly.

- `make test.valgrind` has been adjusted to account for *cpan/dist/ext* separation.

- On compilers that support it, **-Wwrite-strings** is now added to cflags by default.

- The *Encode* module can now (once again) be included in a static Perl build. The special-case handling for this situation got broken in Perl 5.11.0, and has now been repaired.

- The previous default size of a PerlIO buffer (4096 bytes) has been increased to the larger of 8192 bytes and your local BUFSIZ. Benchmarks show that doubling this decade-old default increases read and write performance by around 25% to 50% when using the default layers of perlio on top of unix. To choose a non-default size, such as to get back the old value or to obtain an even larger value, configure with:

  ```
  ./Configure -Accflags=-DPERLIOBUF_DEFAULT_BUFSIZ=N
  ```

  where N is the desired size in bytes; it should probably be a multiple of your page size.

- An "incompatible operand types" error in ternary expressions when building with `clang` has been fixed (5.12.2).

- Perl now skips setuid *File::Copy* tests on partitions it detects mounted as `nosuid` (5.12.2).

Platform Support

New Platforms

AIX

Perl now builds on AIX 4.2 (5.12.1).

Discontinued Platforms

Apollo DomainOS

The last vestiges of support for this platform have been excised from the Perl distribution. It was officially discontinued in version 5.12.0. It had not worked for years before that.

MacOS Classic

The last vestiges of support for this platform have been excised from the Perl distribution. It was officially discontinued in an earlier version.

Platform-Specific Notes

AIX

- *README.aix* has been updated with information about the XL C/C++ V11 compiler suite (5.12.2).

ARM

- The `d_u32align` configuration probe on ARM has been fixed (5.12.2).

Cygwin

- *MakeMaker* has been updated to build manpages on cygwin.
- Improved rebase behaviour

  If a DLL is updated on cygwin the old imagebase address is reused. This solves most rebase errors, especially when updating on core DLL's. See *http://www.tishler.net/jason/software/rebase/rebase-2.4.2.README* for more information.
- Support for the standard cygwin dll prefix (needed for FFIs)
- Updated build hints file

FreeBSD 7

- FreeBSD 7 no longer contains */usr/bin/objformat*. At build time, Perl now skips the *objformat* check for versions 7 and higher and assumes ELF (5.12.1).

HP-UX

- Perl now allows **-Duse64bitint** without promoting to `use64bitall` on HP-UX (5.12.1).

IRIX

- Conversion of strings to floating-point numbers is now more accurate on IRIX systems [perl #32380].

Mac OS X

- Early versions of Mac OS X (Darwin) had buggy implementations of the setregid(), setreuid(), setrgid(,) and setruid() functions, so Perl would pretend they did not exist.

  These functions are now recognised on Mac OS 10.5 (Leopard; Darwin 9) and higher, as they have been fixed [perl #72990].

MirBSD

- Previously if you built Perl with a shared *libperl.so* on MirBSD (the default config), it would work up to the installation; however, once installed, it would be unable to find *libperl*. Path handling is now treated as in the other BSD dialects.

NetBSD

- The NetBSD hints file has been changed to make the system malloc the default.

OpenBSD

- OpenBSD > 3.7 has a new malloc implementation which is *mmap*-based, and as such can release memory back to the OS; however, Perl's use of this malloc causes a substantial slowdown, so we now default to using Perl's malloc instead [perl #75742].

OpenVOS

- Perl now builds again with OpenVOS (formerly known as Stratus VOS) [perl #78132] (5.12.3).

Solaris

- DTrace is now supported on Solaris. There used to be build failures, but these have been fixed [perl #73630] (5.12.3).

VMS

- Extension building on older (pre 7.3-2) VMS systems was broken because configure.com hit the DCL symbol length limit of 1K. We now work within this limit when assembling the list of extensions in the core build (5.12.1).

- We fixed configuring and building Perl with **-Uuseperlio** (5.12.1).
- `PerlIOUnix_open` now honours the default permissions on VMS.

  When `perlio` became the default and `unix` became the default bottom layer, the most common path for creating files from Perl became `PerlIOUnix_open`, which has always explicitly used `0666` as the permission mask. This prevents inheriting permissions from RMS defaults and ACLs, so to avoid that problem, we now pass `0777` to open(). In the VMS CRTL, `0777` has a special meaning over and above intersecting with the current umask; specifically, it allows Unix syscalls to preserve native default permissions (5.12.3).
- The shortening of symbols longer than 31 characters in the core C sources and in extensions is now by default done by the C compiler rather than by xsubpp (which could only do so for generated symbols in XS code). You can reenable xsubpp's symbol shortening by configuring with -Uuseshortenedsymbols, but you'll have some work to do to get the core sources to compile.
- Record-oriented files (record format variable or variable with fixed control) opened for write by the `perlio` layer will now be line-buffered to prevent the introduction of spurious line breaks whenever the perlio buffer fills up.
- *git_version.h* is now installed on VMS. This was an oversight in v5.12.0 which caused some extensions to fail to build (5.12.2).
- Several memory leaks in *stat()* have been fixed (5.12.2).
- A memory leak in Perl_rename() due to a double allocation has been fixed (5.12.2).
- A memory leak in vms_fid_to_name() (used by realpath() and realname()> has been fixed (5.12.2).

Windows

See also *fork() emulation will not wait for signalled children* and *Perl source code is read in text mode on Windows*, above.

- Fixed build process for SDK2003SP1 compilers.
- Compilation with Visual Studio 2010 is now supported.
- When using old 32-bit compilers, the define `_USE_32BIT_TIME_T` is now set in `$Config{ccflags}`. This improves portability when compiling XS extensions using new compilers, but for a Perl compiled with old 32-bit compilers.
- `$Config{gccversion}` is now set correctly when Perl is built using the mingw64 compiler from *http://mingw64.org* [perl #73754].
- When building Perl with the mingw64 x64 cross-compiler `incpath`, `libpth`, `ldflags`, `lddlflags` and `ldflags_nolargefiles` values in *Config.pm* and *Config_heavy.pl* were not previously being set correctly because, with that compiler, the include and lib directories are not immediately below `$(CCHOME)` (5.12.2).
- The build process proceeds more smoothly with mingw and dmake when *C:\MSYS\bin* is in the PATH, due to a `Cwd` fix.
- Support for building with Visual C++ 2010 is now underway, but is not yet complete. See *README.win32* or *perlwin32* for more details.
- The option to use an externally-supplied crypt(), or to build with no crypt() at all, has been removed. Perl supplies its own crypt() implementation for Windows, and the political situation that required this part of the distribution to sometimes be omitted is long gone.

Internal Changes

New APIs

CLONE_PARAMS structure added to ease correct thread creation

Modules that create threads should now create `CLONE_PARAMS` structures by calling the new function Perl_clone_params_new(), and free them with Perl_clone_params_del(). This will ensure compatibility with any future changes to the internals of the `CLONE_PARAMS` structure layout, and that it is correctly allocated and initialised.

New parsing functions

Several functions have been added for parsing Perl statements and expressions. These functions are meant to be used by XS code invoked during Perl parsing, in a recursive-descent manner, to allow modules to augment the standard Perl syntax.

- *parse_stmtseq()* parses a sequence of statements, up to closing brace or EOF.
- *parse_fullstmt()* parses a complete Perl statement, including optional label.
- *parse_barestmt()* parses a statement without a label.
- *parse_block()* parses a code block.
- *parse_label()* parses a statement label, separate from statements.
- *`parse_fullexpr()`*, *`parse_listexpr()`*, *`parse_termexpr()`*, and *`parse_arithexpr()`* parse expressions at various precedence levels.

Hints hash API

A new C API for introspecting the hinthash `%^H` at runtime has been added. See `cop_hints_2hv`, `cop_hints_fetchpvn`, `cop_hints_fetchpvs`, `cop_hints_fetchsv`, and `hv_copy_hints_hv` in *perlapi* for details.

A new, experimental API has been added for accessing the internal structure that Perl uses for `%^H`. See the functions beginning with `cophh_` in *perlapi*.

C interface to caller()

The `caller_cx` function has been added as an XSUB-writer's equivalent of caller(). See *perlapi* for details.

Custom per-subroutine check hooks

XS code in an extension module can now annotate a subroutine (whether implemented in XS or in Perl) so that nominated XS code will be called at compile time (specifically as part of op checking) to change the op tree of that subroutine. The compile-time check function (supplied by the extension module) can implement argument processing that can't be expressed as a prototype, generate customised compile-time warnings, perform constant folding for a pure function, inline a subroutine consisting of sufficiently simple ops, replace the whole call with a custom op, and so on. This was previously all possible by hooking the `entersub` op checker, but the new mechanism makes it easy to tie the hook to a specific subroutine. See *"cv_set_call_checker" in perlapi*.

To help in writing custom check hooks, several subtasks within standard `entersub` op checking have been separated out and exposed in the API.

Improved support for custom OPs

Custom ops can now be registered with the new `custom_op_register` C function and the `XOP` structure. This will make it easier to add new properties of custom ops in the future. Two new properties have been added already, `xop_class` and `xop_peep`.

`xop_class` is one of the OA_*OP constants. It allows *B* and other introspection mechanisms to work with custom ops that aren't BASEOPs. `xop_peep` is a pointer to a function that will be called for ops

of this type from `Perl_rpeep`.

See *"Custom Operators" in perlguts* and *"Custom Operators" in perlapi* for more detail.

The old `PL_custom_op_names`/`PL_custom_op_descs` interface is still supported but discouraged.

Scope hooks

It is now possible for XS code to hook into Perl's lexical scope mechanism at compile time, using the new `Perl_blockhook_register` function. See *"Compile-time scope hooks" in perlguts*.

The recursive part of the peephole optimizer is now hookable

In addition to `PL_peepp`, for hooking into the toplevel peephole optimizer, a `PL_rpeepp` is now available to hook into the optimizer recursing into side-chains of the optree.

New non-magical variants of existing functions

The following functions/macros have been added to the API. The `*_nomg` macros are equivalent to their non-`_nomg` variants, except that they ignore get-magic. Those ending in `_flags` allow one to specify whether get-magic is processed.

```
sv_2bool_flags
SvTRUE_nomg
sv_2nv_flags
SvNV_nomg
sv_cmp_flags
sv_cmp_locale_flags
sv_eq_flags
sv_collxfrm_flags
```

In some of these cases, the non-`_flags` functions have been replaced with wrappers around the new functions.

pv/pvs/sv versions of existing functions

Many functions ending with pvn now have equivalent `pv`/`pvs`/`sv` versions.

List op-building functions

List op-building functions have been added to the API. See *op_append_elem*, *op_append_list*, and *op_prepend_elem* in *perlapi*.

LINKLIST

The *LINKLIST* macro, part of op building that constructs the execution-order op chain, has been added to the API.

Localisation functions

The `save_freeop`, `save_op`, `save_pushi32ptr` and `save_pushptrptr` functions have been added to the API.

Stash names

A stash can now have a list of effective names in addition to its usual name. The first effective name can be accessed via the `HvENAME` macro, which is now the recommended name to use in MRO linearisations (`HvNAME` being a fallback if there is no `HvENAME`).

These names are added and deleted via `hv_ename_add` and `hv_ename_delete`. These two functions are *not* part of the API.

New functions for finding and removing magic

The *`mg_findext()`* and *`sv_unmagicext()`* functions have been added to the API. They allow extension authors to find and remove magic attached to scalars based on both the magic type and the

magic virtual table, similar to how sv_magicext() attaches magic of a certain type and with a given virtual table to a scalar. This eliminates the need for extensions to walk the list of `MAGIC` pointers of an SV to find the magic that belongs to them.

find_rundefsv

This function returns the SV representing `$_`, whether it's lexical or dynamic.

Perl_croak_no_modify

Perl_croak_no_modify() is short-hand for `Perl_croak("%s", PL_no_modify)`.

PERL_STATIC_INLINE define

The `PERL_STATIC_INLINE` define has been added to provide the best-guess incantation to use for static inline functions, if the C compiler supports C99-style static inline. If it doesn't, it'll give a plain `static`.

`HAS_STATIC_INLINE` can be used to check if the compiler actually supports inline functions.

New pv_escape option for hexadecimal escapes

A new option, `PERL_PV_ESCAPE_NONASCII`, has been added to `pv_escape` to dump all characters above ASCII in hexadecimal. Before, one could get all characters as hexadecimal or the Latin1 non-ASCII as octal.

lex_start

`lex_start` has been added to the API, but is considered experimental.

op_scope() and op_lvalue()

The op_scope() and op_lvalue() functions have been added to the API, but are considered experimental.

C API Changes

PERL_POLLUTE has been removed

The option to define `PERL_POLLUTE` to expose older 5.005 symbols for backwards compatibility has been removed. Its use was always discouraged, and MakeMaker contains a more specific escape hatch:

```
perl Makefile.PL POLLUTE=1
```

This can be used for modules that have not been upgraded to 5.6 naming conventions (and really should be completely obsolete by now).

Check API compatibility when loading XS modules

When Perl's API changes in incompatible ways (which usually happens between major releases), XS modules compiled for previous versions of Perl will no longer work. They need to be recompiled against the new Perl.

The `XS_APIVERSION_BOOTCHECK` macro has been added to ensure that modules are recompiled and to prevent users from accidentally loading modules compiled for old perls into newer perls. That macro, which is called when loading every newly compiled extension, compares the API version of the running perl with the version a module has been compiled for and raises an exception if they don't match.

Perl_fetch_cop_label

The first argument of the C API function `Perl_fetch_cop_label` has changed from `struct refcounted_he *` to `COP *`, to insulate the user from implementation details.

This API function was marked as "may change", and likely isn't in use outside the core. (Neither an unpacked CPAN nor Google's codesearch finds any other references to it.)

GvCV() and GvGP() are no longer lvalues

The new GvCV_set() and GvGP_set() macros are now provided to replace assignment to those two macros.

This allows a future commit to eliminate some backref magic between GV and CVs, which will require complete control over assignment to the `gp_cv` slot.

CvGV() is no longer an lvalue

Under some circumstances, the CvGV() field of a CV is now reference-counted. To ensure consistent behaviour, direct assignment to it, for example `CvGV(cv) = gv` is now a compile-time error. A new macro, `CvGV_set(cv,gv)` has been introduced to run this operation safely. Note that modification of this field is not part of the public API, regardless of this new macro (and despite its being listed in this section).

CvSTASH() is no longer an lvalue

The CvSTASH() macro can now only be used as an rvalue. CvSTASH_set() has been added to replace assignment to CvSTASH(). This is to ensure that backreferences are handled properly. These macros are not part of the API.

Calling conventions for newFOROP and newWHILEOP

The way the parser handles labels has been cleaned up and refactored. As a result, the newFOROP() constructor function no longer takes a parameter stating what label is to go in the state op.

The newWHILEOP() and newFOROP() functions no longer accept a line number as a parameter.

Flags passed to uvuni_to_utf8_flags and utf8n_to_uvuni

Some of the flags parameters to uvuni_to_utf8_flags() and utf8n_to_uvuni() have changed. This is a result of Perl's now allowing internal storage and manipulation of code points that are problematic in some situations. Hence, the default actions for these functions has been complemented to allow these code points. The new flags are documented in *perlapi*. Code that requires the problematic code points to be rejected needs to change to use the new flags. Some flag names are retained for backward source compatibility, though they do nothing, as they are now the default. However the flags `UNICODE_ALLOW_FDD0`, `UNICODE_ALLOW_FFFF`, `UNICODE_ILLEGAL`, and `UNICODE_IS_ILLEGAL` have been removed, as they stem from a fundamentally broken model of how the Unicode non-character code points should be handled, which is now described in *"Non-character code points" in perlunicode*. See also the Unicode section under *Selected Bug Fixes*.

Deprecated C APIs

`Perl_ptr_table_clear`

`Perl_ptr_table_clear` is no longer part of Perl's public API. Calling it now generates a deprecation warning, and it will be removed in a future release.

`sv_compile_2op`

The sv_compile_2op() API function is now deprecated. Searches suggest that nothing on CPAN is using it, so this should have zero impact.

It attempted to provide an API to compile code down to an optree, but failed to bind correctly to lexicals in the enclosing scope. It's not possible to fix this problem within the constraints of its parameters and return value.

`find_rundefsvoffset`

The `find_rundefsvoffset` function has been deprecated. It appeared that its design was insufficient for reliably getting the lexical `$_` at run-time.

Use the new `find_rundefsv` function or the `UNDERBAR` macro instead. They directly return the right SV representing `$_`, whether it's lexical or dynamic.

`CALL_FPTR` and `CPERLscope`

Those are left from an old implementation of MULTIPLICITY using C++ objects, which was removed in Perl 5.8. Nowadays these macros do exactly nothing, so they shouldn't be used anymore.

For compatibility, they are still defined for external XS code. Only extensions defining PERL_CORE must be updated now.

Other Internal Changes

Stack unwinding

The protocol for unwinding the C stack at the last stage of a `die` has changed how it identifies the target stack frame. This now uses a separate variable `PL_restartjmpenv`, where previously it relied on the `blk_eval.cur_top_env` pointer in the `eval` context frame that has nominally just been discarded. This change means that code running during various stages of Perl-level unwinding no longer needs to take care to avoid destroying the ghost frame.

Scope stack entries

The format of entries on the scope stack has been changed, resulting in a reduction of memory usage of about 10%. In particular, the memory used by the scope stack to record each active lexical variable has been halved.

Memory allocation for pointer tables

Memory allocation for pointer tables has been changed. Previously `Perl_ptr_table_store` allocated memory from the same arena system as SV bodies and HEs, with freed memory remaining bound to those arenas until interpreter exit. Now it allocates memory from arenas private to the specific pointer table, and that memory is returned to the system when `Perl_ptr_table_free` is called. Additionally, allocation and release are both less CPU intensive.

UNDERBAR

The UNDERBAR macro now calls `find_rundefsv`. dUNDERBAR is now a noop but should still be used to ensure past and future compatibility.

String comparison routines renamed

The `ibcmp_*` functions have been renamed and are now called `foldEQ`, `foldEQ_locale`, and `foldEQ_utf8`. The old names are still available as macros.

chop and chomp implementations merged

The opcode bodies for `chop` and `chomp` and for `schop` and `schomp` have been merged. The implementation functions Perl_do_chop() and Perl_do_chomp(), never part of the public API, have been merged and moved to a static function in *pp.c*. This shrinks the Perl binary slightly, and should not affect any code outside the core (unless it is relying on the order of side-effects when `chomp` is passed a *list* of values).

Selected Bug Fixes

I/O

- Perl no longer produces this warning:

```
$ perl -we 'open(my $f, ">", \my $x); binmode($f, "scalar")'
Use of uninitialized value in binmode at -e line 1.
```

- Opening a glob reference via `open($fh, ">", \*glob)` no longer causes the glob to be corrupted when the filehandle is printed to. This would cause Perl to crash whenever the glob's contents were accessed [perl #77492].

- PerlIO no longer crashes when called recursively, such as from a signal handler. Now it just leaks memory [perl #75556].

- Most I/O functions were not warning for unopened handles unless the "closed" and

"unopened" warnings categories were both enabled. Now only `use warnings 'unopened'` is necessary to trigger these warnings, as had always been the intention.

- There have been several fixes to PerlIO layers:

  When `binmode(FH, ":crlf")` pushes the `:crlf` layer on top of the stack, it no longer enables crlf layers lower in the stack so as to avoid unexpected results [perl #38456].

  Opening a file in `:raw` mode now does what it advertises to do (first open the file, then `binmode` it), instead of simply leaving off the top layer [perl #80764].

  The three layers `:pop`, `:utf8`, and `:bytes` didn't allow stacking when opening a file. For example this:

  ```
  open(FH, ">:pop:perlio", "some.file") or die $!;
  ```

  would throw an "Invalid argument" error. This has been fixed in this release [perl #82484].

Regular Expression Bug Fixes

- The regular expression engine no longer loops when matching `"\N{LATIN SMALL LIGATURE FF}" =~ /f+/i` and similar expressions [perl #72998] (5.12.1).
- The trie runtime code should no longer allocate massive amounts of memory, fixing #74484.
- Syntax errors in `(?{...})` blocks no longer cause panic messages [perl #2353].
- A pattern like `(?:(o){2})?` no longer causes a "panic" error [perl #39233].
- A fatal error in regular expressions containing `(.*?)` when processing UTF-8 data has been fixed [perl #75680] (5.12.2).
- An erroneous regular expression engine optimisation that caused regex verbs like `*COMMIT` sometimes to be ignored has been removed.
- The regular expression bracketed character class `[\8\9]` was effectively the same as `[89\000]`, incorrectly matching a NULL character. It also gave incorrect warnings that the 8 and 9 were ignored. Now `[\8\9]` is the same as `[89]` and gives legitimate warnings that `\8` and `\9` are unrecognized escape sequences, passed-through.
- A regular expression match in the right-hand side of a global substitution (`s///g`) that is in the same scope will no longer cause match variables to have the wrong values on subsequent iterations. This can happen when an array or hash subscript is interpolated in the right-hand side, as in `s|(.)|@a{ print($1), /./ }|g` [perl #19078].
- Several cases in which characters in the Latin-1 non-ASCII range (0x80 to 0xFF) used not to match themselves, or used to match both a character class and its complement, have been fixed. For instance, U+00E2 could match both `\w` and `\W` [perl #78464] [perl #18281] [perl #60156].
- Matching a Unicode character against an alternation containing characters that happened to match continuation bytes in the former's UTF8 representation (like `qq{\x{30ab}} =~ /\xab|\xa9/`) would cause erroneous warnings [perl #70998].
- The trie optimisation was not taking empty groups into account, preventing "foo" from matching `/\A(?:(?:)foo|bar|zot)\z/` [perl #78356].
- A pattern containing a + inside a lookahead would sometimes cause an incorrect match failure in a global match (for example, `/(?=(\S+))/g`) [perl #68564].
- A regular expression optimisation would sometimes cause a match with a `{n,m}` quantifier to fail when it should have matched [perl #79152].
- Case-insensitive matching in regular expressions compiled under `use locale` now works

much more sanely when the pattern or target string is internally encoded in UTF8. Previously, under these conditions the localeness was completely lost. Now, code points above 255 are treated as Unicode, but code points between 0 and 255 are treated using the current locale rules, regardless of whether the pattern or the string is encoded in UTF8. The few case-insensitive matches that cross the 255/256 boundary are not allowed. For example, 0xFF does not caselessly match the character at 0x178, LATIN CAPITAL LETTER Y WITH DIAERESIS, because 0xFF may not be LATIN SMALL LETTER Y in the current locale, and Perl has no way of knowing if that character even exists in the locale, much less what code point it is.

- The `(?|...)` regular expression construct no longer crashes if the final branch has more sets of capturing parentheses than any other branch. This was fixed in Perl 5.10.1 for the case of a single branch, but that fix did not take multiple branches into account [perl #84746].
- A bug has been fixed in the implementation of `{...}` quantifiers in regular expressions that prevented the code block in `/((\w+)(?{ print $2 })){2}/` from seeing the `$2` sometimes [perl #84294].

Syntax/Parsing Bugs

- `when (scalar) {...}` no longer crashes, but produces a syntax error [perl #74114] (5.12.1).
- A label right before a string eval (`foo: eval $string`) no longer causes the label to be associated also with the first statement inside the eval [perl #74290] (5.12.1).
- The `no 5.13.2` form of `no` no longer tries to turn on features or pragmata (like *strict*) [perl #70075] (5.12.2).
- `BEGIN {require 5.12.0}` now behaves as documented, rather than behaving identically to `use 5.12.0`. Previously, `require` in a `BEGIN` block was erroneously executing the `use feature ':5.12.0'` and `use strict` behaviour, which only `use` was documented to provide [perl #69050].
- A regression introduced in Perl 5.12.0, making `my $x = 3; $x = length(undef)` result in `$x` set to 3 has been fixed. `$x` will now be `undef` [perl #85508] (5.12.2).
- When strict "refs" mode is off, `%{...}` in rvalue context returns `undef` if its argument is undefined. An optimisation introduced in Perl 5.12.0 to make `keys %{...}` faster when used as a boolean did not take this into account, causing `keys %{+undef}` (and `keys %$foo` when `$foo` is undefined) to be an error, which it should be so in strict mode only [perl #81750].
- Constant-folding used to cause

  ```
  $text =~ ( 1 ? /phoo/ : /bear/)
  ```

  to turn into

  ```
  $text =~ /phoo/
  ```

  at compile time. Now it correctly matches against `$_` [perl #20444].
- Parsing Perl code (either with string `eval` or by loading modules) from within a `UNITCHECK` block no longer causes the interpreter to crash [perl #70614].
- String `eval`s no longer fail after 2 billion scopes have been compiled [perl #83364].
- The parser no longer hangs when encountering certain Unicode characters, such as U+387 [perl #74022].
- Defining a constant with the same name as one of Perl's special blocks (like `INIT`) stopped

working in 5.12.0, but has now been fixed [perl #78634].

- A reference to a literal value used as a hash key (`$hash{\"foo"}`) used to be stringified, even if the hash was tied [perl #79178].
- A closure containing an `if` statement followed by a constant or variable is no longer treated as a constant [perl #63540].
- `state` can now be used with attributes. It used to mean the same thing as `my` if any attributes were present [perl #68658].
- Expressions like `@$a > 3` no longer cause $a to be mentioned in the "Use of uninitialized value in numeric gt" warning when $a is undefined (since it is not part of the > expression, but the operand of the @) [perl #72090].
- Accessing an element of a package array with a hard-coded number (as opposed to an arbitrary expression) would crash if the array did not exist. Usually the array would be autovivified during compilation, but typeglob manipulation could remove it, as in these two cases which used to crash:

```
*d = *a;  print $d[0];
undef *d; print $d[0];
```

- The **-C** command-line option, when used on the shebang line, can now be followed by other options [perl #72434].
- The `B` module was returning `B::OP`s instead of `B::LOGOP`s for `entertry` [perl #80622]. This was due to a bug in the Perl core, not in `B` itself.

Stashes, Globs and Method Lookup

Perl 5.10.0 introduced a new internal mechanism for caching MROs (method resolution orders, or lists of parent classes; aka "isa" caches) to make method lookup faster (so `@ISA` arrays would not have to be searched repeatedly). Unfortunately, this brought with it quite a few bugs. Almost all of these have been fixed now, along with a few MRO-related bugs that existed before 5.10.0:

- The following used to have erratic effects on method resolution, because the "isa" caches were not reset or otherwise ended up listing the wrong classes. These have been fixed.

  Aliasing packages by assigning to globs [perl #77358]

  Deleting packages by deleting their containing stash elements

  Undefining the glob containing a package (`undef *Foo::`)

  Undefining an ISA glob (`undef *Foo::ISA`)

  Deleting an ISA stash element (`delete $Foo::{ISA}`)

  Sharing @ISA arrays between classes (via `*Foo::ISA = \@Bar::ISA` or `*Foo::ISA = *Bar::ISA`) [perl #77238]

  `undef *Foo::ISA` would even stop a new `@Foo::ISA` array from updating caches.

- Typeglob assignments would crash if the glob's stash no longer existed, so long as the glob assigned to were named `ISA` or the glob on either side of the assignment contained a subroutine.
- `PL_isarev`, which is accessible to Perl via `mro::get_isarev` is now updated properly when packages are deleted or removed from the `@ISA` of other classes. This allows many packages to be created and deleted without causing a memory leak [perl #75176].

In addition, various other bugs related to typeglobs and stashes have been fixed:

- Some work has been done on the internal pointers that link between symbol tables (stashes),

typeglobs, and subroutines. This has the effect that various edge cases related to deleting stashes or stash entries (for example, <%FOO:: = ()>), and complex typeglob or code-reference aliasing, will no longer crash the interpreter.

- Assigning a reference to a glob copy now assigns to a glob slot instead of overwriting the glob with a scalar [perl #1804] [perl #77508].

- A bug when replacing the glob of a loop variable within the loop has been fixed [perl #21469]. This means the following code will no longer crash:

```
for $x (...) {
    *x = *y;
}
```

- Assigning a glob to a PVLV used to convert it to a plain string. Now it works correctly, and a PVLV can hold a glob. This would happen when a nonexistent hash or array element was passed to a subroutine:

```
sub { $_[0] = *foo }->($hash{key});
# $_[0] would have been the string "*main::foo"
```

  It also happened when a glob was assigned to, or returned from, an element of a tied array or hash [perl #36051].

- When trying to report `Use of uninitialized value $Foo::BAR`, crashes could occur if the glob holding the global variable in question had been detached from its original stash by, for example, `delete $::{"Foo::"}`. This has been fixed by disabling the reporting of variable names in those cases.

- During the restoration of a localised typeglob on scope exit, any destructors called as a result would be able to see the typeglob in an inconsistent state, containing freed entries, which could result in a crash. This would affect code like this:

```
local *@;
eval { die bless [] }; # puts an object in $@
sub DESTROY {
  local $@; # boom
}
```

  Now the glob entries are cleared before any destructors are called. This also means that destructors can vivify entries in the glob. So Perl tries again and, if the entries are re-created too many times, dies with a "panic: gp_free ..." error message.

- If a typeglob is freed while a subroutine attached to it is still referenced elsewhere, the subroutine is renamed to `__ANON__` in the same package, unless the package has been undefined, in which case the `__ANON__` package is used. This could cause packages to be sometimes autovivified, such as if the package had been deleted. Now this no longer occurs. The `__ANON__` package is also now used when the original package is no longer attached to the symbol table. This avoids memory leaks in some cases [perl #87664].

- Subroutines and package variables inside a package whose name ends with `::` can now be accessed with a fully qualified name.

Unicode

- What has become known as "the Unicode Bug" is almost completely resolved in this release. Under `use feature 'unicode_strings'` (which is automatically selected by `use 5.012` and above), the internal storage format of a string no longer affects the external semantics. [perl #58182].

  There are two known exceptions:

1 The now-deprecated, user-defined case-changing functions require utf8-encoded strings to operate. The CPAN module *Unicode::Casing* has been written to replace this feature without its drawbacks, and the feature is scheduled to be removed in 5.16.

2 quotemeta() (and its in-line equivalent `\Q`) can also give different results depending on whether a string is encoded in UTF-8. See *"The "Unicode Bug"" in perlunicode*.

- Handling of Unicode non-character code points has changed. Previously they were mostly considered illegal, except that in some place only one of the 66 of them was known. The Unicode Standard considers them all legal, but forbids their "open interchange". This is part of the change to allow internal use of any code point (see *Core Enhancements*). Together, these changes resolve [perl #38722], [perl #51918], [perl #51936], and [perl #63446].
- Case-insensitive `"/i"` regular expression matching of Unicode characters that match multiple characters now works much more as intended. For example

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /ffi/ui
  ```

  and

  ```
  "ffi" =~ /\N{LATIN SMALL LIGATURE FFI}/ui
  ```

  are both true. Previously, there were many bugs with this feature. What hasn't been fixed are the places where the pattern contains the multiple characters, but the characters are split up by other things, such as in

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /(f)(f)i/ui
  ```

  or

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /ffi*/ui
  ```

  or

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /[a-f][f-m][g-z]/ui
  ```

  None of these match.

  Also, this matching doesn't fully conform to the current Unicode Standard, which asks that the matching be made upon the NFD (Normalization Form Decomposed) of the text. However, as of this writing (April 2010), the Unicode Standard is currently in flux about what they will recommend doing with regard in such scenarios. It may be that they will throw out the whole concept of multi-character matches. [perl #71736].
- Naming a deprecated character in `\N{`*NAME*`}` no longer leaks memory.
- We fixed a bug that could cause `\N{`*NAME*`}` constructs followed by a single `"."` to be parsed incorrectly [perl #74978] (5.12.1).
- `chop` now correctly handles characters above `"\x{7fffffff}"` [perl #73246].
- Passing to `index` an offset beyond the end of the string when the string is encoded internally in UTF8 no longer causes panics [perl #75898].
- warn() and die() now respect utf8-encoded scalars [perl #45549].
- Sometimes the UTF8 length cache would not be reset on a value returned by substr, causing `length(substr($uni_string, ...))` to give wrong answers. With `${^UTF8CACHE}` set to -1, it would also produce a "panic" error message [perl #77692].

Ties, Overloading and Other Magic

- Overloading now works properly in conjunction with tied variables. What formerly happened was that most ops checked their arguments for overloading *before* checking for magic, so for example an overloaded object returned by a tied array access would usually be treated as not overloaded [RT #57012].
- Various instances of magic (like tie methods) being called on tied variables too many or too few times have been fixed:
  - `$tied->()` did not always call FETCH [perl #8438].
  - Filetest operators and `y///` and `tr///` were calling FETCH too many times.
  - The = operator used to ignore magic on its right-hand side if the scalar happened to hold a typeglob (if a typeglob was the last thing returned from or assigned to a tied scalar) [perl #77498].
  - Dereference operators used to ignore magic if the argument was a reference already (such as from a previous FETCH) [perl #72144].
  - `splice` now calls set-magic (so changes made by `splice @ISA` are respected by method calls) [perl #78400].
  - In-memory files created by `open($fh, ">", \$buffer)` were not calling FETCH/STORE at all [perl #43789] (5.12.2).
  - utf8::is_utf8() now respects get-magic (like `$1`) (5.12.1).
- Non-commutative binary operators used to swap their operands if the same tied scalar was used for both operands and returned a different value for each FETCH. For instance, if `$t` returned 2 the first time and 3 the second, then `$t/$t` would evaluate to 1.5. This has been fixed [perl #87708].
- String `eval` now detects taintedness of overloaded or tied arguments [perl #75716].
- String `eval` and regular expression matches against objects with string overloading no longer cause memory corruption or crashes [perl #77084].
- *readline* now honors <> overloading on tied arguments.
- `<expr>` always respects overloading now if the expression is overloaded.

  Because "<> as glob" was parsed differently from "<> as filehandle" from 5.6 onwards, something like `<$foo[0]>` did not handle overloading, even if `$foo[0]` was an overloaded object. This was contrary to the documentation for *overload*, and meant that <> could not be used as a general overloaded iterator operator.
- The fallback behaviour of overloading on binary operators was asymmetric [perl #71286].
- Magic applied to variables in the main package no longer affects other packages. See *Magic variables outside the main package* above [perl #76138].
- Sometimes magic (ties, taintedness, etc.) attached to variables could cause an object to last longer than it should, or cause a crash if a tied variable were freed from within a tie method. These have been fixed [perl #81230].
- DESTROY methods of objects implementing ties are no longer able to crash by accessing the tied variable through a weak reference [perl #86328].
- Fixed a regression of kill() when a match variable is used for the process ID to kill [perl #75812].
- `$AUTOLOAD` used to remain tainted forever if it ever became tainted. Now it is correctly

untainted if an autoloaded method is called and the method name was not tainted.

- `sprintf` now dies when passed a tainted scalar for the format. It did already die for arbitrary expressions, but not for simple scalars [perl #82250].
- `lc`, `uc`, `lcfirst`, and `ucfirst` no longer return untainted strings when the argument is tainted. This has been broken since perl 5.8.9 [perl #87336].

The Debugger

- The Perl debugger now also works in taint mode [perl #76872].
- Subroutine redefinition works once more in the debugger [perl #48332].
- When **-d** is used on the shebang (`#!`) line, the debugger now has access to the lines of the main program. In the past, this sometimes worked and sometimes did not, depending on the order in which things happened to be arranged in memory [perl #71806].
- A possible memory leak when using *caller()* to set `@DB::args` has been fixed (5.12.2).
- Perl no longer stomps on `$DB::single`, `$DB::trace`, and `$DB::signal` if these variables already have values when `$^P` is assigned to [perl #72422].
- `#line` directives in string evals were not properly updating the arrays of lines of code (`@{"_< ..."}`) that the debugger (or any debugging or profiling module) uses. In threaded builds, they were not being updated at all. In non-threaded builds, the line number was ignored, so any change to the existing line number would cause the lines to be misnumbered [perl #79442].

Threads

- Perl no longer accidentally clones lexicals in scope within active stack frames in the parent when creating a child thread [perl #73086].
- Several memory leaks in cloning and freeing threaded Perl interpreters have been fixed [perl #77352].
- Creating a new thread when directory handles were open used to cause a crash, because the handles were not cloned, but simply passed to the new thread, resulting in a double free.

  Now directory handles are cloned properly on Windows and on systems that have a `fchdir` function. On other systems, new threads simply do not inherit directory handles from their parent threads [perl #75154].
- The typeglob `*,`, which holds the scalar variable `$,` (output field separator), had the wrong reference count in child threads.
- [perl #78494] When pipes are shared between threads, the `close` function (and any implicit close, such as on thread exit) no longer blocks.
- Perl now does a timely cleanup of SVs that are cloned into a new thread but then discovered to be orphaned (that is, their owners are *not* cloned). This eliminates several "scalars leaked" warnings when joining threads.

Scoping and Subroutines

- Lvalue subroutines are again able to return copy-on-write scalars. This had been broken since version 5.10.0 [perl #75656] (5.12.3).
- `require` no longer causes `caller` to return the wrong file name for the scope that called `require` and other scopes higher up that had the same file name [perl #68712].
- `sort` with a `($$)`-prototyped comparison routine used to cause the value of `@_` to leak out of the sort. Taking a reference to `@_` within the sorting routine could cause a crash [perl #72334].

- Match variables (like $1) no longer persist between calls to a sort subroutine [perl #76026].
- Iterating with `foreach` over an array returned by an lvalue sub now works [perl #23790].
- $@ is now localised during calls to `binmode` to prevent action at a distance [perl #78844].
- Calling a closure prototype (what is passed to an attribute handler for a closure) now results in a "Closure prototype called" error message instead of a crash [perl #68560].
- Mentioning a read-only lexical variable from the enclosing scope in a string `eval` no longer causes the variable to become writable [perl #19135].

Signals

- Within signal handlers, `$!` is now implicitly localized.
- CHLD signals are no longer unblocked after a signal handler is called if they were blocked before by `POSIX::sigprocmask` [perl #82040].
- A signal handler called within a signal handler could cause leaks or double-frees. Now fixed [perl #76248].

Miscellaneous Memory Leaks

- Several memory leaks when loading XS modules were fixed (5.12.2).
- *substr()*, *pos()*, *keys()*, and *vec()* could, when used in combination with lvalues, result in leaking the scalar value they operate on, and cause its destruction to happen too late. This has now been fixed.
- The postincrement and postdecrement operators, ++ and `--`, used to cause leaks when used on references. This has now been fixed.
- Nested `map` and `grep` blocks no longer leak memory when processing large lists [perl #48004].
- `use` *`VERSION`* and `no` *`VERSION`* no longer leak memory [perl #78436] [perl #69050].
- `.=` followed by <> or `readline` would leak memory if `$/` contained characters beyond the octet range and the scalar assigned to happened to be encoded as UTF8 internally [perl #72246].
- `eval 'BEGIN{die}'` no longer leaks memory on non-threaded builds.

Memory Corruption and Crashes

- glob() no longer crashes when `%File::Glob::` is empty and `CORE::GLOBAL::glob` isn't present [perl #75464] (5.12.2).
- readline() has been fixed when interrupted by signals so it no longer returns the "same thing" as before or random memory.
- When assigning a list with duplicated keys to a hash, the assignment used to return garbage and/or freed values:

  ```
  @a = %h = (list with some duplicate keys);
  ```

  This has now been fixed [perl #31865].
- The mechanism for freeing objects in globs used to leave dangling pointers to freed SVs, meaning Perl users could see corrupted state during destruction.

  Perl now frees only the affected slots of the GV, rather than freeing the GV itself. This makes sure that there are no dangling refs or corrupted state during destruction.
- The interpreter no longer crashes when freeing deeply-nested arrays of arrays. Hashes have

not been fixed yet [perl #44225].

- Concatenating long strings under `use encoding` no longer causes Perl to crash [perl #78674].
- Calling `->import` on a class lacking an import method could corrupt the stack, resulting in strange behaviour. For instance,

  ```
  push @a, "foo", $b = bar->import;
  ```

  would assign "foo" to $b [perl #63790].
- The `recv` function could crash when called with the MSG_TRUNC flag [perl #75082].
- `formline` no longer crashes when passed a tainted format picture. It also taints `$^A` now if its arguments are tainted [perl #79138].
- A bug in how we process filetest operations could cause a segfault. Filetests don't always expect an op on the stack, so we now use TOPs only if we're sure that we're not `stat`ing the _ filehandle. This is indicated by `OPf_KIDS` (as checked in ck_ftst) [perl #74542] (5.12.1).
- unpack() now handles scalar context correctly for `%32H` and `%32u`, fixing a potential crash. split() would crash because the third item on the stack wasn't the regular expression it expected. `unpack("%2H", ...)` would return both the unpacked result and the checksum on the stack, as would `unpack("%2u", ...)` [perl #73814] (5.12.2).

Fixes to Various Perl Operators

- The `&`, `|`, and `^` bitwise operators no longer coerce read-only arguments [perl #20661].
- Stringifying a scalar containing "-0.0" no longer has the effect of turning false into true [perl #45133].
- Some numeric operators were converting integers to floating point, resulting in loss of precision on 64-bit platforms [perl #77456].
- sprintf() was ignoring locales when called with constant arguments [perl #78632].
- Combining the vector (`%v`) flag and dynamic precision would cause `sprintf` to confuse the order of its arguments, making it treat the string as the precision and vice-versa [perl #83194].

Bugs Relating to the C API

- The C-level `lex_stuff_pvn` function would sometimes cause a spurious syntax error on the last line of the file if it lacked a final semicolon [perl #74006] (5.12.1).
- The `eval_sv` and `eval_pv` C functions now set $@ correctly when there is a syntax error and no `G_KEEPERR` flag, and never set it if the `G_KEEPERR` flag is present [perl #3719].
- The XS multicall API no longer causes subroutines to lose reference counts if called via the multicall interface from within those very subroutines. This affects modules like *List::Util*. Calling one of its functions with an active subroutine as the first argument could cause a crash [perl #78070].
- The `SvPVbyte` function available to XS modules now calls magic before downgrading the SV, to avoid warnings about wide characters [perl #72398].
- The ref types in the typemap for XS bindings now support magical variables [perl #72684].
- `sv_catsv_flags` no longer calls `mg_get` on its second argument (the source string) if the flags passed to it do not include SV_GMAGIC. So it now matches the documentation.
- `my_strftime` no longer leaks memory. This fixes a memory leak in `POSIX::strftime` [perl #73520].

- *XSUB.h* now correctly redefines fgets under PERL_IMPLICIT_SYS [perl #55049] (5.12.1).
- XS code using fputc() or fputs() on Windows could cause an error due to their arguments being swapped [perl #72704] (5.12.1).
- A possible segfault in the `T_PTROBJ` default typemap has been fixed (5.12.2).
- A bug that could cause "Unknown error" messages when `call_sv(code, G_EVAL)` is called from an XS destructor has been fixed (5.12.2).

Known Problems

This is a list of significant unresolved issues which are regressions from earlier versions of Perl or which affect widely-used CPAN modules.

- `List::Util::first` misbehaves in the presence of a lexical `$_` (typically introduced by `my $_` or implicitly by `given`). The variable that gets set for each iteration is the package variable `$_`, not the lexical `$_`.

  A similar issue may occur in other modules that provide functions which take a block as their first argument, like

  ```
  foo { ... $_ ...} list
  ```

  See also: *http://rt.perl.org/rt3/Public/Bug/Display.html?id=67694*

- readline() returns an empty string instead of a cached previous value when it is interrupted by a signal
- The changes in prototype handling break *Switch*. A patch has been sent upstream and will hopefully appear on CPAN soon.
- The upgrade to *ExtUtils-MakeMaker-6.57_05* has caused some tests in the *Module-Install* distribution on CPAN to fail. (Specifically, *02_mymeta.t* tests 5 and 21; *18_all_from.t* tests 6 and 15; *19_authors.t* tests 5, 13, 21, and 29; and *20_authors_with_special_characters.t* tests 6, 15, and 23 in version 1.00 of that distribution now fail.)
- On VMS, `Time::HiRes` tests will fail due to a bug in the CRTL's implementation of `setitimer`: previous timer values would be cleared if a timer expired but not if the timer was reset before expiring. HP OpenVMS Engineering have corrected the problem and will release a patch in due course (Quix case # QXCM1001115136).
- On VMS, there were a handful of `Module::Build` test failures we didn't get to before the release; please watch CPAN for updates.

Errata

keys(), values(), and each() work on arrays

You can now use the keys(), values(), and each() builtins on arrays; previously you could use them only on hashes. See *perlfunc* for details. This is actually a change introduced in perl 5.12.0, but it was missed from that release's *perl5120delta*.

split() and @_

split() no longer modifies `@_` when called in scalar or void context. In void context it now produces a "Useless use of split" warning. This was also a perl 5.12.0 change that missed the perldelta.

Obituary

Randy Kobes, creator of http://kobesearch.cpan.org/ and contributor/maintainer to several core Perl toolchain modules, passed away on September 18, 2010 after a battle with lung cancer. The community was richer for his involvement. He will be missed.

Acknowledgements

Perl 5.14.0 represents one year of development since Perl 5.12.0 and contains nearly 550,000 lines of changes across nearly 3,000 files from 150 authors and committers.

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 5.14.0:

Aaron Crane, Abhijit Menon-Sen, Abigail, Ã†var ArnfjÃ¶rÃ° Bjarmason, Alastair Douglas, Alexander Alekseev, Alexander Hartmaier, Alexandr Ciornii, Alex Davies, Alex Vandiver, Ali Polatel, Allen Smith, Andreas KÃ¶nig, Andrew Rodland, Andy Armstrong, Andy Dougherty, Aristotle Pagaltzis, Arkturuz, Arvan, A. Sinan Unur, Ben Morrow, Bo Lindbergh, Boris Ratner, Brad Gilbert, Bram, brian d foy, Brian Phillips, Casey West, Charles Bailey, Chas. Owens, Chip Salzenberg, Chris 'BinGOs' Williams, chromatic, Craig A. Berry, Curtis Jewell, Dagfinn Ilmari MannsÃ¥ker, Dan Dascalescu, Dave Rolsky, David Caldwell, David Cantrell, David Golden, David Leadbeater, David Mitchell, David Wheeler, Eric Brine, Father Chrysostomos, Fingle Nark, Florian Ragwitz, Frank Wiegand, Franz Fasching, Gene Sullivan, George Greer, Gerard Goossen, Gisle Aas, Goro Fuji, Grant McLean, gregor herrmann, H.Merijn Brand, Hongwen Qiu, Hugo van der Sanden, Ian Goodacre, James E Keenan, James Mastros, Jan Dubois, Jay Hannah, Jerry D. Hedden, Jesse Vincent, Jim Cromie, Jirka HruÅ¡ka, John Peacock, Joshua ben Jore, Joshua Pritikin, Karl Williamson, Kevin Ryde, kmx, Lars DÉªá´‡á´„á´‹á´•á´¡ è¿ªæ‹‰æ–¯, Larwan Berke, Leon Brocard, Leon Timmermans, Lubomir Rintel, Lukas Mai, Maik Hentsche, Marty Pauley, Marvin Humphrey, Matt Johnson, Matt S Trout, Max Maischein, Michael Breen, Michael Fig, Michael G Schwern, Michael Parker, Michael Stevens, Michael Witten, Mike Kelly, Moritz Lenz, Nicholas Clark, Nick Cleaton, Nick Johnston, Nicolas Kaiser, Niko Tyni, Noirin Shirley, Nuno Carvalho, Paul Evans, Paul Green, Paul Johnson, Paul Marquess, Peter J. Holzer, Peter John Acklam, Peter Martini, Philippe Bruhat (BooK), Piotr Fusik, Rafael Garcia-Suarez, Rainer Tammer, Reini Urban, Renee Baecker, Ricardo Signes, Richard MÃ¶hn, Richard Soderberg, Rob Hoelz, Robin Barker, Ruslan Zakirov, Salvador FandiÃ±o, Salvador Ortiz Garcia, Shlomi Fish, Sinan Unur, Sisyphus, Slaven Rezic, Steffen MÃ¼ller, Steve Hay, Steven Schubiger, Steve Peters, Sullivan Beck, Tatsuhiko Miyagawa, Tim Bunce, Todd Rinaldo, Tom Christiansen, Tom Hukins, Tony Cook, Tye McQueen, Vadim Konovalov, Vernon Lyon, Vincent Pit, Walt Mankowski, Wolfram Humann, Yves Orton, Zefram, and ZsbÃ¡n Ambrus.

This is woefully incomplete as it's automatically generated from version control history. In particular, it doesn't include the names of the (very much appreciated) contributors who reported issues in previous versions of Perl that helped make Perl 5.14.0 better. For a more complete list of all of Perl's historical contributors, please see the AUTHORS file in the Perl 5.14.0 distribution.

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.

Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the Perl bug database at http://rt.perl.org/perlbug/ . 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 are 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 use this address for security issues in the Perl core *only*, 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.