NAME

perl5110delta - what is new for perl v5.11.0

DESCRIPTION

This document describes differences between the 5.10.0 release and the 5.11.0 development release.

Incompatible Changes

Unicode interpretation of \w, \d, \s, and the POSIX character classes redefined.

Previous versions of Perl tried to map POSIX style character class definitions onto Unicode property names so that patterns would "dwim" when matches were made against latin-1 or unicode strings. This proved to be a mistake, breaking character class negation, causing forward compatibility problems (as Unicode keeps updating their property definitions and adding new characters), and other problems.

Therefore we have now defined a new set of artificial "unicode" property names which will be used to do unicode matching of patterns using POSIX style character classes and perl short-form escape character classes like \w and \d.

The key change here is that \d will no longer match every digit in the unicode standard (there are thousands) nor will \w match every word character in the standard, instead they will match precisely their POSIX or Perl definition.

Those needing to match based on Unicode properties can continue to do so by using the \p{} syntax to match whichever property they like, including the new artificial definitions.

NOTE: This is a backwards incompatible no-warning change in behaviour. If you are upgrading and you process large volumes of text look for POSIX and Perl style character classes and change them to the relevant property name (by removing the word 'Posix' from the current name).

The following table maps the POSIX character class names, the escapes and the old and new Unicode property mappings:

```
POSIX  Esc  Class               New-Property  ! Old-Property
-----------------------------------------------+-------------
alnum       [0-9A-Za-z]         IsPosixAlnum  ! IsAlnum
alpha       [A-Za-z]            IsPosixAlpha  ! IsAlpha
ascii       [\000-\177]         IsASCII       = IsASCII
blank       [\011 ]             IsPosixBlank  !
cntrl       [\0-\37\177]        IsPosixCntrl  ! IsCntrl
digit   \d  [0-9]               IsPosixDigit  ! IsDigit
graph       [!-~]               IsPosixGraph  ! IsGraph
lower       [a-z]               IsPosixLower  ! IsLower
print       [ -~]               IsPosixPrint  ! IsPrint
punct       [!-/:-@[-`{-~]      IsPosixPunct  ! IsPunct
space       [\11-\15 ]          IsPosixSpace  ! IsSpace
        \s  [\11\12\14\15 ]     IsPerlSpace   ! IsSpacePerl
upper       [A-Z]               IsPosixUpper  ! IsUpper
word    \w  [0-9A-Z_a-z]        IsPerlWord    ! IsWord
xdigit      [0-9A-Fa-f]         IsXDigit      = IsXDigit
```

If you wish to build perl with the old mapping you may do so by setting

```
 #define PERL_LEGACY_UNICODE_CHARCLASS_MAPPINGS 1
```

in regcomp.h, and then setting

```
PERL_TEST_LEGACY_POSIX_CC
```

to true your environment when testing.

@INC reorganization

In @INC, ARCHLIB and PRIVLIB now occur after after the current version's site_perl and vendor_perl.

Switch statement changes

The handling of complex expressions by the `given`/`when` switch statement has been enhanced. These enhancements are also available in 5.10.1 and subsequent 5.10 releases. There are two new cases where `when` now interprets its argument as a boolean, instead of an expression to be used in a smart match:

flip-flop operators

The `..` and `...` flip-flop operators are now evaluated in boolean context, following their usual semantics; see *"Range Operators" in perlop*.

Note that, as in perl 5.10.0, `when (1..10)` will not work to test whether a given value is an integer between 1 and 10; you should use `when ([1..10])` instead (note the array reference).

However, contrary to 5.10.0, evaluating the flip-flop operators in boolean context ensures it can now be useful in a `when()`, notably for implementing bistable conditions, like in:

```
when (/^=begin/ .. /^=end/) {
  # do something
}
```

defined-or operator

A compound expression involving the defined-or operator, as in `when (expr1 // expr2)`, will be treated as boolean if the first expression is boolean. (This just extends the existing rule that applies to the regular or operator, as in `when (expr1 || expr2)`.)

The next section details more changes brought to the semantics to the smart match operator, that naturally also modify the behaviour of the switch statements where smart matching is implicitly used. These changers were also made for the 5.10.1 release, and will remain in subsequent 5.10 releases.

Smart match changes

Changes to type-based dispatch

The smart match operator ~~ is no longer commutative. The behaviour of a smart match now depends primarily on the type of its right hand argument. Moreover, its semantics have been adjusted for greater consistency or usefulness in several cases. While the general backwards compatibility is maintained, several changes must be noted:

- Code references with an empty prototype are no longer treated specially. They are passed an argument like the other code references (even if they choose to ignore it).
- `%hash ~~ sub {}` and `@array ~~ sub {}` now test that the subroutine returns a true value for each key of the hash (or element of the array), instead of passing the whole hash or array as a reference to the subroutine.
- Due to the commutativity breakage, code references are no longer treated specially when appearing on the left of the ~~ operator, but like any vulgar scalar.
- `undef ~~ %hash` is always false (since `undef` can't be a key in a hash). No implicit conversion to `""` is done (as was the case in perl 5.10.0).
- `$scalar ~~ @array` now always distributes the smart match across the elements of the

array. It's true if one element in @array verifies `$scalar ~~ $element`. This is a generalization of the old behaviour that tested whether the array contained the scalar.

The full dispatch table for the smart match operator is given in *"Smart matching in detail" in perlsyn*.

Smart match and overloading

According to the rule of dispatch based on the rightmost argument type, when an object overloading ~~ appears on the right side of the operator, the overload routine will always be called (with a 3rd argument set to a true value, see *overload*.) However, when the object will appear on the left, the overload routine will be called only when the rightmost argument is a simple scalar. This way distributivity of smart match across arrays is not broken, as well as the other behaviours with complex types (coderefs, hashes, regexes). Thus, writers of overloading routines for smart match mostly need to worry only with comparing against a scalar, and possibly with stringification overloading; the other common cases will be automatically handled consistently.

~~ will now refuse to work on objects that do not overload it (in order to avoid relying on the object's underlying structure). (However, if the object overloads the stringification or the numification operators, and if overload fallback is active, it will be used instead, as usual.)

Labels can't be keywords

Labels used as targets for the `goto`, `last`, `next` or `redo` statements cannot be keywords anymore. This restriction will prevent potential confusion between the `goto LABEL` and `goto EXPR` syntaxes: for example, a statement like `goto print` would jump to a label whose name would be the return value of `print()`, (usually 1), instead of a label named `print`. Moreover, the other control flow statements would just ignore any keyword passed to them as a label name. Since such labels cannot be defined anymore, this kind of error will be avoided.

Other incompatible changes

- The semantics of `use feature :5.10*` have changed slightly. See *Modules and Pragmata* for more information.
- It is now a run-time error to use the smart match operator ~~ with an object that has no overload defined for it. (This way ~~ will not break encapsulation by matching against the object's internal representation as a reference.)
- The version control system used for the development of the perl interpreter has been switched from Perforce to git. This is mainly an internal issue that only affects people actively working on the perl core; but it may have minor external visibility, for example in some of details of the output of `perl -V`. See *perlrepository* for more information.
- The internal structure of the `ext/` directory in the perl source has been reorganised. In general, a module `Foo::Bar` whose source was stored under *ext/Foo/Bar/* is now located under *ext/Foo-Bar/*. Also, nearly all dual-life modules have been moved from *lib/* to *ext/*. This is purely a source tarball change, and should make no difference to the compilation or installation of perl, unless you have a very customised build process that explicitly relies on this structure, or which hard-codes the `nonxs_ext` *Configure* parameter. Specifically, this change does not by default alter the location of any files in the final installation.
- As part of the `Test::Harness` 2.x to 3.x upgrade, the experimental `Test::Harness::Straps` module has been removed. See *Updated Modules* for more details.
- As part of the `ExtUtils::MakeMaker` upgrade, the `ExtUtils::MakeMaker::bytes` and `ExtUtils::MakeMaker::vmsish` modules have been removed from this distribution.
- `Module::CoreList` no longer contains the `%:patchlevel` hash.
- This one is actually a change introduced in 5.10.0, but it was missed from that release's perldelta, so it is mentioned here instead.

A bugfix related to the handling of the `/m` modifier and `qr` resulted in a change of behaviour between 5.8.x and 5.10.0:

```
# matches in 5.8.x, doesn't match in 5.10.0
$re = qr/^bar/; "foo\nbar" =~ /$re/m;
```

- `length undef` now returns undef.
- Unsupported private C API functions are now declared "static" to prevent leakage to Perl's public API.
- To support the bootstrapping process, *miniperl* no longer builds with UTF-8 support in the regexp engine.

  This allows a build to complete with PERL_UNICODE set and a UTF-8 locale. Without this there's a bootstrapping problem, as miniperl can't load the UTF-8 components of the regexp engine, because they're not yet built.
- *miniperl*'s @INC is now restricted to just -I..., the split of $ENV{PERL5LIB}, and "."
- A space or a newline is now required after a "`#line XXX`" directive.
- Tied filehandles now have an additional method EOF which provides the EOF type
- To better match all other flow control statements, `foreach` may no longer be used as an attribute.

Core Enhancements

Unicode Character Database 5.1.0

The copy of the Unicode Character Database included in Perl 5.11.0 has been updated to 5.1.0 from 5.0.0. See *http://www.unicode.org/versions/Unicode5.1.0/#Notable_Changes* for the notable changes.

A proper interface for pluggable Method Resolution Orders

As of Perl 5.11.0 there is a new interface for plugging and using method resolution orders other than the default (linear depth first search). The C3 method resolution order added in 5.10.0 has been re-implemented as a plugin, without changing its Perl-space interface. See *perlmroapi* for more information.

The overloading pragma

This pragma allows you to lexically disable or enable overloading for some or all operations. (Yuval Kogman)

\N regex escape

A new regex escape has been added, `\N`. It will match any character that is not a newline, independently from the presence or absence of the single line match modifier `/s`. (If `\N` is followed by an opening brace and by a letter, perl will still assume that a Unicode character name is coming, so compatibility is preserved.) (Rafael Garcia-Suarez)

Implicit strictures

Using the `use VERSION` syntax with a version number greater or equal to 5.11.0 will also lexically enable strictures just like `use strict` would do (in addition to enabling features.) So, the following:

```
use 5.11.0;
```

will now imply:

```
use strict;
use feature ':5.11';
```

Parallel tests

The core distribution can now run its regression tests in parallel on Unix-like platforms. Instead of running `make test`, set `TEST_JOBS` in your environment to the number of tests to run in parallel, and run `make test_harness`. On a Bourne-like shell, this can be done as

```
TEST_JOBS=3 make test_harness  # Run 3 tests in parallel
```

An environment variable is used, rather than parallel make itself, because *TAP::Harness* needs to be able to schedule individual non-conflicting test scripts itself, and there is no standard interface to `make` utilities to interact with their job schedulers.

Note that currently some test scripts may fail when run in parallel (most notably `ext/IO/t/io_dir.t`). If necessary run just the failing scripts again sequentially and see if the failures go away.

The ... operator

A new operator, `...`, nicknamed the Yada Yada operator, has been added. It is intended to mark placeholder code, that is not yet implemented. See *"Yada Yada Operator" in perlop*. (chromatic)

DTrace support

Some support for DTrace has been added. See "DTrace support" in *INSTALL*.

Support for configure_requires in CPAN module metadata

Both `CPAN` and `CPANPLUS` now support the `configure_requires` keyword in the *META.yml* metadata file included in most recent CPAN distributions. This allows distribution authors to specify configuration prerequisites that must be installed before running *Makefile.PL* or *Build.PL*.

See the documentation for `ExtUtils::MakeMaker` or `Module::Build` for more on how to specify `configure_requires` when creating a distribution for CPAN.

each is now more flexible

The `each` function can now operate on arrays.

Y2038 compliance

Perl's core time-related functions are now Y2038 compliant. (With 29 years to spare!)

$, flexibility

The variable `$,` may now be tied.

// in where clauses

// now behaves like || in when clauses

Enabling warnings from your shell environment

You can now set `-W` from the `PERL5OPT` environment variable

delete local

`delete local` now allows you to locally delete a hash entry.

New support for Abstract namespace sockets

Abstract namespace sockets are Linux-specific socket type that live in AF_UNIX family, slightly abusing it to be able to use arbitrary character arrays as addresses: They start with nul byte and are not terminated by nul byte, but with the length passed to the socket() system call.

Modules and Pragmata

Dual-lifed modules moved

Dual-lifed modules maintained primarily in the Perl core now live in dist/. Dual-lifed modules maintained primarily on CPAN now live in cpan/

In previous releases of Perl, it was customary to enumerate all module changes in this section of the `perldelta` file. From 5.11.0 forward only notable updates (such as new or deprecated modules) will be listed in this section. For a complete reference to the versions of modules shipped in a given release of perl, please see *Module::CoreList*.

New Modules and Pragmata

`autodie`

This is a new lexically-scoped alternative for the `Fatal` module. The bundled version is 2.06_01. Note that in this release, using a string eval when `autodie` is in effect can cause the autodie behaviour to leak into the surrounding scope. See *"BUGS" in autodie* for more details.

`Compress::Raw::Bzip2`

This has been added to the core (version 2.020).

`parent`

This pragma establishes an ISA relationship with base classes at compile time. It provides the key feature of `base` without the feature creep.

`Parse::CPAN::Meta`

This has been added to the core (version 1.39).

Pragmata Changes

`overloading`

See *The `overloading` pragma* above.

`attrs`

The `attrs` pragma has been removed. It had been marked as deprecated since 5.6.0.

`charnames`

The Unicode *NameAliases.txt* database file has been added. This has the effect of adding some extra `\N` character names that formerly wouldn't have been recognised; for example, `"\N{LATIN CAPITAL LETTER GHA}"`.

`feature`

The meaning of the `:5.10` and `:5.10.X` feature bundles has changed slightly. The last component, if any (i.e. `X`) is simply ignored. This is predicated on the assumption that new features will not, in general, be added to maintenance releases. So `:5.10` and `:5.10.X` have identical effect. This is a change to the behaviour documented for 5.10.0.

`mro`

Upgraded from version 1.00 to 1.01. Performance for single inheritance is 40% faster - see *Performance Enhancements* below.

`mro` is now implemented as an XS extension. The documented interface has not changed. Code relying on the implementation detail that some `mro::` methods happened to be available at all times gets to "keep both pieces".

Updated Modules

`ExtUtils::MakeMaker`

Upgraded from version 6.42 to 6.55_02.

Note that `ExtUtils::MakeMaker::bytes` and `ExtUtils::MakeMaker::vmsish` have

been removed from this distribution.

`Test::Harness`

Upgraded from version 2.64 to 3.17.

Note that one side-effect of the 2.x to 3.x upgrade is that the experimental `Test::Harness::Straps` module (and its supporting `Assert`, `Iterator`, `Point` and `Results` modules) have been removed. If you still need this, then they are available in the (unmaintained) `Test-Harness-Straps` distribution on CPAN.

`UNIVERSAL`

Upgraded from version 1.04 to 1.05.

`UNIVERSAL->import()` is now deprecated.

Utility Changes

h2ph

Now looks in `include-fixed` too, which is a recent addition to gcc's search path.

h2xs

No longer incorrectly treats enum values like macros (Daniel Burr).

Now handles C++ style constants (`//`) properly in enums. (A patch from Rainer Weikusat was used; Daniel Burr also proposed a similar fix).

perl5db.pl

`LVALUE` subroutines now work under the debugger.

The debugger now correctly handles proxy constant subroutines, and subroutine stubs.

perlbug

perlbug now uses `%Module::CoreList::bug_tracker` to print out upstream bug tracker URLs.

Where the user names a module that their bug report is about, and we know the URL for its upstream bug tracker, provide a message to the user explaining that the core copies the CPAN version directly, and provide the URL for reporting the bug directly to upstream.

perlthanks

Perl 5.11.0 added a new utility *perlthanks*, which is a variant of *perlbug*, but for sending non-bug-reports to the authors and maintainers of Perl. Getting nothing but bug reports can become a bit demoralising: we'll see if this changes things.

New Documentation

perlhaiku

This contains instructions on how to build perl for the Haiku platform.

perlmroapi

This describes the new interface for pluggable Method Resolution Orders.

perlperf

This document, by Richard Foley, provides an introduction to the use of performance and optimization techniques which can be used with particular reference to perl programs.

perlrepository

This describes how to access the perl source using the *git* version control system.

Changes to Existing Documentation

The various large *Changes** files (which listed every change made to perl over the last 18 years) have been removed, and replaced by a small file, also called *Changes*, which just explains how that same information may be extracted from the git version control system.

The file *Porting/patching.pod* has been deleted, as it mainly described interacting with the old Perforce-based repository, which is now obsolete. Information still relevant has been moved to *perlrepository*.

perlapi, *perlintern*, *perlmodlib* and *perltoc* are now all generated at build time, rather than being shipped as part of the release.

- Documented -X overloading.
- Documented that `when()` treats specially most of the filetest operators
- Documented when as a syntax modifier
- Eliminated "Old Perl threads tutorial", which describes 5005 threads.

  pod/perlthrtut.pod is the same material reworked for ithreads.
- Correct previous documentation: v-strings are not deprecated

  With version objects, we need them to use MODULE VERSION syntax. This patch removes the deprecation note.
- Added security contact information to *perlsec*

Performance Enhancements

- A new internal cache means that `isa()` will often be faster.
- The implementation of `C3` Method Resolution Order has been optimised - linearisation for classes with single inheritance is 40% faster. Performance for multiple inheritance is unchanged.
- Under `use locale`, the locale-relevant information is now cached on read-only values, such as the list returned by `keys %hash`. This makes operations such as `sort keys %hash` in the scope of `use locale` much faster.
- Empty `DESTROY` methods are no longer called.
- Faster `Perl_sv_utf8_upgrade()`
- Speed up `keys` on empty hash

Installation and Configuration Improvements

ext/ reorganisation

The layout of directories in *ext* has been revised. Specifically, all extensions are now flat, and at the top level, with `/` in pathnames replaced by `-`, so that *ext/Data/Dumper/* is now *ext/Data-Dumper/*, etc. The names of the extensions as specified to *Configure*, and as reported by `%Config::Config` under the keys `dynamic_ext`, `known_extensions`, `nonxs_ext` and `static_ext` have not changed, and still use `/`. Hence this change will not have any affect once perl is installed. `Safe` has been split out from being part of `Opcode`, and `mro` is now an extension in its own right.

Nearly all dual-life modules have been moved from *lib* to *ext*, and will now appear as known `nonxs_ext`. This will made no difference to the structure of an installed perl, nor will the modules installed differ, unless you run *Configure* with options to specify an exact list of extensions to build. In this case, you will rapidly become aware that you need to add to your list, because various modules needed to complete the build, such as `ExtUtils::ParseXS`, have now become extensions, and without them the build will fail well before it attempts to run the regression tests.

Configuration improvements

If `vendorlib` and `vendorarch` are the same, then they are only added to `@INC` once.

`$Config{usedevel}` and the C-level `PERL_USE_DEVEL` are now defined if perl is built with `-Dusedevel`.

Configure will enable use of `-fstack-protector`, to provide protection against stack-smashing attacks, if the compiler supports it.

Configure will now determine the correct prototypes for re-entrant functions, and for `gconvert`, if you are using a C++ compiler rather than a C compiler.

On Unix, if you build from a tree containing a git repository, the configuration process will note the commit hash you have checked out, for display in the output of `perl -v` and `perl -V`. Unpushed local commits are automatically added to the list of local patches displayed by `perl -V`.

Compilation improvements

As part of the flattening of *ext*, all extensions on all platforms are built by *make_ext.pl*. This replaces the Unix-specific *ext/util/make_ext*, VMS-specific *make_ext.com* and Win32-specific *win32/buildext.pl*.

Platform Specific Changes

AIX

Removed *libbsd* for AIX 5L and 6.1. Only `flock()` was used from *libbsd*.

Removed *libgdbm* for AIX 5L and 6.1. The *libgdbm* is delivered as an optional package with the AIX Toolbox. Unfortunately the 64 bit version is broken.

Hints changes mean that AIX 4.2 should work again.

Cygwin

On Cygwin we now strip the last number from the DLL. This has been the behaviour in the cygwin.com build for years. The hints files have been updated.

DomainOS

Support for Apollo DomainOS was removed in Perl 5.11.0

FreeBSD

The hints files now identify the correct threading libraries on FreeBSD 7 and later.

Irix

We now work around a bizarre preprocessor bug in the Irix 6.5 compiler: `cc -E -` unfortunately goes into K&R mode, but `cc -E file.c` doesn't.

Haiku

Patches from the Haiku maintainers have been merged in. Perl should now build on Haiku.

MachTen

Support for Tenon Intersystems MachTen Unix layer for MacOS Classic was removed in Perl 5.11.0

MiNT

Support for Atari MiNT was removed in Perl 5.11.0.

MirOS BSD

Perl should now build on MirOS BSD.

NetBSD

Hints now supports versions 5.*.

Stratus VOS

Various changes from Stratus have been merged in.

Symbian

There is now support for Symbian S60 3.2 SDK and S60 5.0 SDK.

Win32

Improved message window handling means that `alarm` and `kill` messages will no longer be dropped under race conditions.

VMS

Reads from the in-memory temporary files of `PerlIO::scalar` used to fail if `$/` was set to a numeric reference (to indicate record-style reads). This is now fixed.

VMS now supports `getgrgid`.

Many improvements and cleanups have been made to the VMS file name handling and conversion code.

Enabling the `PERL_VMS_POSIX_EXIT` logical name now encodes a POSIX exit status in a VMS condition value for better interaction with GNV's bash shell and other utilities that depend on POSIX exit values. See *"$?" in perlvms* for details.

`File::Copy` now detects Unix compatibility mode on VMS.

Selected Bug Fixes

- `-I` on shebang line now adds directories in front of @INC as documented, and as does `-I` when specified on the command-line.
- `kill` is now fatal when called on non-numeric process identifiers. Previously, an 'undef' process identifier would be interpreted as a request to kill process "0", which would terminate the current process group on POSIX systems. Since process identifiers are always integers, killing a non-numeric process is now fatal.
- 5.10.0 inadvertently disabled an optimisation, which caused a measurable performance drop in list assignment, such as is often used to assign function parameters from `@_`. The optimisation has been re-instated, and the performance regression fixed.
- Fixed memory leak on `while (1) { map 1, 1 }` [RT #53038].
- Some potential coredumps in PerlIO fixed [RT #57322,54828].
- The debugger now works with lvalue subroutines.
- The debugger's `m` command was broken on modules that defined constants [RT #61222].
- `crypt` and string complement could return tainted values for untainted arguments [RT #59998].
- The `-i`*suffix* command-line switch now recreates the file using restricted permissions, before changing its mode to match the original file. This eliminates a potential race condition [RT #60904].
- On some Unix systems, the value in $? would not have the top bit set (`$? & 128`) even if the child core dumped.
- Under some circumstances, `$^R` could incorrectly become undefined [RT #57042].
- In the XS API, various hash functions, when passed a pre-computed hash where the key is UTF-8, might result in an incorrect lookup.
- XS code including *XSUB.h* before *perl.h* gave a compile-time error [RT #57176].

- `$object->isa('Foo')` would report false if the package `Foo` didn't exist, even if the object's `@ISA` contained `Foo`.
- Various bugs in the new-to 5.10.0 mro code, triggered by manipulating `@ISA`, have been found and fixed.
- Bitwise operations on references could crash the interpreter, e.g. `$x=\$y; $x |= "foo"` [RT #54956].
- Patterns including alternation might be sensitive to the internal UTF-8 representation, e.g.

```
my $byte = chr(192);
my $utf8 = chr(192); utf8::upgrade($utf8);
$utf8 =~ /$byte|X}/i; # failed in 5.10.0
```

- Within UTF8-encoded Perl source files (i.e. where `use utf8` is in effect), double-quoted literal strings could be corrupted where a `\xNN`, `\0NNN` or `\N{}` is followed by a literal character with ordinal value greater than 255 [RT #59908].
- `B::Deparse` failed to correctly deparse various constructs: `readpipe STRING` [RT #62428], `CORE::require(STRING)` [RT #62488], `sub foo(_)` [RT #62484].
- Using `setpgrp` with no arguments could corrupt the perl stack.
- The block form of `eval` is now specifically trappable by `Safe` and `ops`. Previously it was erroneously treated like string `eval`.
- In 5.10.0, the two characters `[~` were sometimes parsed as the smart match operator (`~~`) [RT #63854].
- In 5.10.0, the `*` quantifier in patterns was sometimes treated as `{0,32767}` [RT #60034, #60464]. For example, this match would fail:

```
("ab" x 32768) =~ /^(ab)*$/
```

- `shmget` was limited to a 32 bit segment size on a 64 bit OS [RT #63924].
- Using `next` or `last` to exit a `given` block no longer produces a spurious warning like the following:

```
Exiting given via last at foo.pl line 123
```

- On Windows, `'.\foo'` and `'..\foo'` were treated differently than `'./foo'` and `'../foo'` by `do` and `require` [RT #63492].
- Assigning a format to a glob could corrupt the format; e.g.:

```
*bar=*foo{FORMAT}; # foo format now bad
```

- Attempting to coerce a typeglob to a string or number could cause an assertion failure. The correct error message is now generated, `Can't coerce GLOB to` *`$type`*.
- Under `use filetest 'access'`, `-x` was using the wrong access mode. This has been fixed [RT #49003].
- `length` on a tied scalar that returned a Unicode value would not be correct the first time. This has been fixed.
- Using an array `tie` inside in array `tie` could SEGV. This has been fixed. [RT #51636]
- A race condition inside `PerlIOStdio_close()` has been identified and fixed. This used to cause various threading issues, including SEGVs.

- In `unpack`, the use of `()` groups in scalar context was internally placing a list on the interpreter's stack, which manifested in various ways, including SEGVs. This is now fixed [RT #50256].
- Magic was called twice in `substr`, `\&$x`, `tie $x, $m` and `chop`. These have all been fixed.
- A 5.10.0 optimisation to clear the temporary stack within the implicit loop of `s///ge` has been reverted, as it turned out to be the cause of obscure bugs in seemingly unrelated parts of the interpreter [commit ef0d4e17921ee3de].
- The line numbers for warnings inside `elsif` are now correct.
- The `..` operator now works correctly with ranges whose ends are at or close to the values of the smallest and largest integers.
- `binmode STDIN, ':raw'` could lead to segmentation faults on some platforms. This has been fixed [RT #54828].
- An off-by-one error meant that `index $str, ...` was effectively being executed as `index "$str\0", ...`. This has been fixed [RT #53746].
- Various leaks associated with named captures in regexes have been fixed [RT #57024].
- A weak reference to a hash would leak. This was affecting `DBI` [RT #56908].
- Using (?|) in a regex could cause a segfault [RT #59734].
- Use of a UTF-8 `tr//` within a closure could cause a segfault [RT #61520].
- Calling `Perl_sv_chop()` or otherwise upgrading an SV could result in an unaligned 64-bit access on the SPARC architecture [RT #60574].
- In the 5.10.0 release, `inc_version_list` would incorrectly list `5.10.*` after `5.8.*`; this affected the `@INC` search order [RT #67628].
- In 5.10.0, `pack "a*", $tainted_value` returned a non-tainted value [RT #52552].
- In 5.10.0, `printf` and `sprintf` could produce the fatal error `panic: utf8_mg_pos_cache_update` when printing UTF-8 strings [RT #62666].
- In the 5.10.0 release, a dynamically created `AUTOLOAD` method might be missed (method cache issue) [RT #60220,60232].
- In the 5.10.0 release, a combination of `use feature` and `//ee` could cause a memory leak [RT #63110].
- `-C` on the shebang (`#!`) line is once more permitted if it is also specified on the command line. `-C` on the shebang line used to be a silent no-op *if* it was not also on the command line, so perl 5.10.0 disallowed it, which broke some scripts. Now perl checks whether it is also on the command line and only dies if it is not [RT #67880].
- In 5.10.0, certain types of re-entrant regular expression could crash, or cause the following assertion failure [RT #60508]:

  ```
  Assertion rx->sublen >= (s - rx->subbeg) + i failed
  ```

- Previously missing files from Unicode 5.1 Character Database are now included.
- `TMPDIR` is now honored when opening an anonymous temporary file

New or Changed Diagnostics

```
panic: sv_chop %s
```

This new fatal error occurs when the C routine `Perl_sv_chop()` was passed a position that is not within the scalar's string buffer. This could be caused by buggy XS code, and at this point recovery is not possible.

`Can't locate package %s for the parents of %s`

This warning has been removed. In general, it only got produced in conjunction with other warnings, and removing it allowed an ISA lookup optimisation to be added.

`v-string in use/require is non-portable`

This warning has been removed.

`Deep recursion on subroutine "%s"`

It is now possible to change the depth threshold for this warning from the default of 100, by recompiling the *perl* binary, setting the C pre-processor macro `PERL_SUB_DEPTH_WARN` to the desired value.

Changed Internals

- TODO: `SVt_RV` is gone. RVs are now stored in IVs
- TODO: REGEXPs are first class
- TODO: OOK is reworked, such that an OOKed scalar is PV not PVIV
- The J.R.R. Tolkien quotes at the head of C source file have been checked and proper citations added, thanks to a patch from Tom Christiansen.
- `Perl_vcroak()` now accepts a null first argument. In addition, a full audit was made of the "not NULL" compiler annotations, and those for several other internal functions were corrected.
- New macros `dSAVEDERRNO`, `dSAVE_ERRNO`, `SAVE_ERRNO`, `RESTORE_ERRNO` have been added to formalise the temporary saving of the `errno` variable.
- The function `Perl_sv_insert_flags` has been added to augment `Perl_sv_insert`.
- The function `Perl_newSV_type(type)` has been added, equivalent to `Perl_newSV()` followed by `Perl_sv_upgrade(type)`.
- The function `Perl_newSVpvn_flags()` has been added, equivalent to `Perl_newSVpvn()` and then performing the action relevant to the flag.

  Two flag bits are currently supported.

  `SVf_UTF8`

  This will call `SvUTF8_on()` for you. (Note that this does not convert an sequence of ISO 8859-1 characters to UTF-8). A wrapper, `newSVpvn_utf8()` is available for this.

  `SVs_TEMP`

  Call `Perl_sv_2mortal()` on the new SV.

  There is also a wrapper that takes constant strings, `newSVpvs_flags()`.
- The function `Perl_croak_xs_usage` has been added as a wrapper to `Perl_croak`.
- The functions `PerlIO_find_layer` and `PerlIO_list_alloc` are now exported.
- `PL_na` has been exterminated from the core code, replaced by local STRLEN temporaries, or `*_nolen()` calls. Either approach is faster than `PL_na`, which is a pointer deference into the interpreter structure under ithreads, and a global variable otherwise.
- `Perl_mg_free()` used to leave freed memory accessible via `SvMAGIC()` on the scalar. It

now updates the linked list to remove each piece of magic as it is freed.

- Under ithreads, the regex in `PL_reg_curpm` is now reference counted. This eliminates a lot of hackish workarounds to cope with it not being reference counted.
- `Perl_mg_magical()` would sometimes incorrectly turn on `SvRMAGICAL()`. This has been fixed.
- The *public* IV and NV flags are now not set if the string value has trailing "garbage". This behaviour is consistent with not setting the public IV or NV flags if the value is out of range for the type.
- SV allocation tracing has been added to the diagnostics enabled by `-Dm`. The tracing can alternatively output via the `PERL_MEM_LOG` mechanism, if that was enabled when the *perl* binary was compiled.
- Smartmatch resolution tracing has been added as a new diagnostic. Use `-DM` to enable it.
- A new debugging flag `-DB` now dumps subroutine definitions, leaving `-Dx` for its original purpose of dumping syntax trees.
- Uses of `Nullav`, `Nullcv`, `Nullhv`, `Nullop`, `Nullsv` etc have been replaced by `NULL` in the core code, and non-dual-life modules, as `NULL` is clearer to those unfamiliar with the core code.
- A macro `MUTABLE_PTR(p)` has been added, which on (non-pedantic) gcc will not cast away `const`, returning a `void *`. Macros `MUTABLE_SV(av)`, `MUTABLE_SV(cv)` etc build on this, casting to `AV *` etc without casting away `const`. This allows proper compile-time auditing of `const` correctness in the core, and helped picked up some errors (now fixed).
- Macros `mPUSHs()` and `mXPUSHs()` have been added, for pushing SVs on the stack and mortalizing them.
- Use of the private structure `mro_meta` has changed slightly. Nothing outside the core should be accessing this directly anyway.
- A new tool, *Porting/expand-macro.pl* has been added, that allows you to view how a C preprocessor macro would be expanded when compiled. This is handy when trying to decode the macro hell that is the perl guts.

New Tests

Many modules updated from CPAN incorporate new tests.

Several tests that have the potential to hang forever if they fail now incorporate a "watchdog" functionality that will kill them after a timeout, which helps ensure that `make test` and `make test_harness` run to completion automatically. (Jerry Hedden).

Some core-specific tests have been added:

t/comp/retainedlines.t

Check that the debugger can retain source lines from `eval`.

t/io/perlio_fail.t

Check that bad layers fail.

t/io/perlio_leaks.t

Check that PerlIO layers are not leaking.

t/io/perlio_open.t

Check that certain special forms of open work.

t/io/perlio.t

General PerlIO tests.

t/io/pvbm.t

Check that there is no unexpected interaction between the internal types PVBM and PVGV.

t/mro/package_aliases.t

Check that mro works properly in the presence of aliased packages.

t/op/dbm.t

Tests for `dbmopen` and `dbmclose`.

t/op/index_thr.t

Tests for the interaction of `index` and threads.

t/op/pat_thr.t

Tests for the interaction of esoteric patterns and threads.

t/op/qr_gc.t

Test that `qr` doesn't leak.

t/op/reg_email_thr.t

Tests for the interaction of regex recursion and threads.

t/op/regexp_qr_embed_thr.t

Tests for the interaction of patterns with embedded `qr//` and threads.

t/op/regexp_unicode_prop.t

Tests for Unicode properties in regular expressions.

t/op/regexp_unicode_prop_thr.t

Tests for the interaction of Unicode properties and threads.

t/op/reg_nc_tie.t

Test the tied methods of `Tie::Hash::NamedCapture`.

t/op/reg_posixcc.t

Check that POSIX character classes behave consistently.

t/op/re.t

Check that exportable `re` functions in *universal.c* work.

t/op/setpgrpstack.t

Check that `setpgrp` works.

t/op/substr_thr.t

Tests for the interaction of `substr` and threads.

t/op/upgrade.t

Check that upgrading and assigning scalars works.

t/uni/lex_utf8.t

Check that Unicode in the lexer works.

t/uni/tie.t

Check that Unicode and `tie` work.

Known Problems

This is a list of some significant unfixed bugs, which are regressions from either 5.10.0 or 5.8.x.

- `List::Util::first` misbehaves in the presence of a lexical $_ (typically introduced by `my $_` or implicitly by `given`). The variable which gets set for each iteration is the package variable $_, not the lexical $_ [RT #67694].

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

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

- The `charnames` pragma may generate a run-time error when a regex is interpolated [RT #56444]:

  ```
  use charnames ':full';
  my $r1 = qr/\N{THAI CHARACTER SARA I}/;
  "foo" =~ $r1;    # okay
  "foo" =~ /$r1+/; # runtime error
  ```

  A workaround is to generate the character outside of the regex:

  ```
  my $a = "\N{THAI CHARACTER SARA I}";
  my $r1 = qr/$a/;
  ```

- Some regexes may run much more slowly when run in a child thread compared with the thread the pattern was compiled into [RT #55600].

Deprecations

The following items are now deprecated.

- `Switch` is buggy and should be avoided. From perl 5.11.0 onwards, it is intended that any use of the core version of this module will emit a warning, and that the module will eventually be removed from the core (probably in perl 5.14.0). See *"Switch statements" in perlsyn* for its replacement.

- The following modules will be removed from the core distribution in a future release, and should be installed from CPAN instead. Distributions on CPAN which require these should add them to their prerequisites. The core versions of these modules warnings will issue a deprecation warning.

  - `Class::ISA`
  - `Pod::Plainer`
  - `Shell`

  Currently support to install from CPAN without a *force* is `TODO` in CPAN and CPANPLUS. This will be addressed before 5.12.0 ships.

- `suidperl` has been removed. It used to provide a mechanism to emulate setuid permission bits on systems that don't support it properly.

- Deprecate assignment to $[

- Remove attrs, which has been deprecated since 1999/10/02.

- Deprecate use of the attribute :locked on subroutines.

- Deprecate using "locked" with the attributes pragma.

- Deprecate using "unique" with the attributes pragma.

- warn if ++ or -- are unable to change the value because it's beyond the limit of representation

  This uses a new warnings category: "imprecision".

- Make lc/uc/lcfirst/ucfirst warn when passed undef.

- Show constant in "Useless use of a constant in void context"

- Make the new warning report undef constants as undef

- Add a new warning, "Prototype after '%s'"

- Tweak the "Illegal character in prototype" warning so it's more precise when reporting illegal characters after _

- Unintended interpolation of $\ in regex

- Make overflow warnings in gmtime/localtime only occur when warnings are on

- Improve mro merging error messages.

  They are now very similar to those produced by Algorithm::C3.

- Amelioration of the error message "Unrecognized character %s in column %d"

  Changes the error message to "Unrecognized character %s; marked by <-- HERE after %s<-- HERE near column %d". This should make it a little simpler to spot and correct the suspicious character.

- Explicitly point to $. when it causes an uninitialized warning for ranges in scalar context

- Deprecated numerous Perl 4-era libraries:

  termcap.pl, *tainted.pl*, *stat.pl*, *shellwords.pl*, *pwd.pl*, *open3.pl*, *open2.pl*, *newgetopt.pl*, *look.pl*, *find.pl*, *finddepth.pl*, *importenv.pl*, *hostname.pl*, *getopts.pl*, *getopt.pl*, *getcwd.pl*, *flush.pl*, *fastcwd.pl*, *exceptions.pl*, *ctime.pl*, *complete.pl*, *cacheout.pl*, *bigrat.pl*, *bigint.pl*, *bigfloat.pl*, *assert.pl*, *abbrev.pl*, *dotsh.pl*, and *timelocal.pl* are all now deprecated. Using them will incur a warning.

Acknowledgements

Some of the work in this release was funded by a TPF grant funded by Dijkmat BV, The Netherlands.

Steffen Mueller and David Golden in particular helped getting CPAN modules polished and synchronised with their in-core equivalents.

Craig Berry was tireless in getting maint to run under VMS, no matter how many times we broke it for him.

The other core committers contributed most of the changes, and applied most of the patches sent in by the hundreds of contributors listed in *AUTHORS*.

Much of the work of categorizing changes in this perldelta file was contributed by the following porters using changelogger.bestpractical.com:

Nicholas Clark, leon, shawn, alexm, rjbs, rafl, Pedro Melo, brunorc, anonymous, â˜„, Tom Hukins, anonymous, Jesse, dagolden, Moritz Onken, Mark Fowler, chorny, anonymous, tmtm

Finally, thanks to Larry Wall, without whom none of this would be necessary.

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 be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

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