NAME

perl58delta - what is new for perl v5.8.0

DESCRIPTION

This document describes differences between the 5.6.0 release and the 5.8.0 release.

Many of the bug fixes in 5.8.0 were already seen in the 5.6.1 maintenance release since the two releases were kept closely coordinated (while 5.8.0 was still called 5.7.something).

Changes that were integrated into the 5.6.1 release are marked `[561]`. Many of these changes have been further developed since 5.6.1 was released, those are marked `[561+]`.

You can see the list of changes in the 5.6.1 release (both from the 5.005_03 release and the 5.6.0 release) by reading *perl561delta*.

Highlights In 5.8.0

- Better Unicode support
- New IO Implementation
- New Thread Implementation
- Better Numeric Accuracy
- Safe Signals
- Many New Modules
- More Extensive Regression Testing

Incompatible Changes

Binary Incompatibility

Perl 5.8 is not binary compatible with earlier releases of Perl.

You have to recompile your XS modules.

(Pure Perl modules should continue to work.)

The major reason for the discontinuity is the new IO architecture called PerlIO. PerlIO is the default configuration because without it many new features of Perl 5.8 cannot be used. In other words: you just have to recompile your modules containing XS code, sorry about that.

In future releases of Perl, non-PerlIO aware XS modules may become completely unsupported. This shouldn't be too difficult for module authors, however: PerlIO has been designed as a drop-in replacement (at the source code level) for the stdio interface.

Depending on your platform, there are also other reasons why we decided to break binary compatibility, please read on.

64-bit platforms and malloc

If your pointers are 64 bits wide, the Perl malloc is no longer being used because it does not work well with 8-byte pointers. Also, usually the system mallocs on such platforms are much better optimized for such large memory models than the Perl malloc. Some memory-hungry Perl applications like the PDL don't work well with Perl's malloc. Finally, other applications than Perl (such as mod_perl) tend to prefer the system malloc. Such platforms include Alpha and 64-bit HPPA, MIPS, PPC, and Sparc.

AIX Dynaloading

The AIX dynaloading now uses in AIX releases 4.3 and newer the native dlopen interface of AIX instead of the old emulated interface. This change will probably break backward compatibility with compiled modules. The change was made to make Perl more compliant with other applications like

mod_perl which are using the AIX native interface.

Attributes for my variables now handled at run-time

The `my EXPR : ATTRS` syntax now applies variable attributes at run-time. (Subroutine and `our` variables still get attributes applied at compile-time.) See *attributes* for additional details. In particular, however, this allows variable attributes to be useful for `tie` interfaces, which was a deficiency of earlier releases. Note that the new semantics doesn't work with the Attribute::Handlers module (as of version 0.76).

Socket Extension Dynamic in VMS

The Socket extension is now dynamically loaded instead of being statically built in. This may or may not be a problem with ancient TCP/IP stacks of VMS: we do not know since we weren't able to test Perl in such configurations.

IEEE-format Floating Point Default on OpenVMS Alpha

Perl now uses IEEE format (T_FLOAT) as the default internal floating point format on OpenVMS Alpha, potentially breaking binary compatibility with external libraries or existing data. G_FLOAT is still available as a configuration option. The default on VAX (D_FLOAT) has not changed.

New Unicode Semantics (no more use utf8, almost)

Previously in Perl 5.6 to use Unicode one would say "use utf8" and then the operations (like string concatenation) were Unicode-aware in that lexical scope.

This was found to be an inconvenient interface, and in Perl 5.8 the Unicode model has completely changed: now the "Unicodeness" is bound to the data itself, and for most of the time "use utf8" is not needed at all. The only remaining use of "use utf8" is when the Perl script itself has been written in the UTF-8 encoding of Unicode. (UTF-8 has not been made the default since there are many Perl scripts out there that are using various national eight-bit character sets, which would be illegal in UTF-8.)

See *perluniintro* for the explanation of the current model, and *utf8* for the current use of the utf8 pragma.

New Unicode Properties

Unicode *scripts* are now supported. Scripts are similar to (and superior to) Unicode *blocks*. The difference between scripts and blocks is that scripts are the glyphs used by a language or a group of languages, while the blocks are more artificial groupings of (mostly) 256 characters based on the Unicode numbering.

In general, scripts are more inclusive, but not universally so. For example, while the script `Latin` includes all the Latin characters and their various diacritic-adorned versions, it does not include the various punctuation or digits (since they are not solely `Latin`).

A number of other properties are now supported, including `\p{L&}`, `\p{Any}` `\p{Assigned}`, `\p{Unassigned}`, `\p{Blank}` [561] and `\p{SpacePerl}` [561] (along with their `\P{...}` versions, of course). See *perlunicode* for details, and more additions.

The `In` or `Is` prefix to names used with the `\p{...}` and `\P{...}` are now almost always optional. The only exception is that a `In` prefix is required to signify a Unicode block when a block name conflicts with a script name. For example, `\p{Tibetan}` refers to the script, while `\p{InTibetan}` refers to the block. When there is no name conflict, you can omit the `In` from the block name (e.g. `\p{BraillePatterns}`), but to be safe, it's probably best to always use the `In`).

REF(...) Instead Of SCALAR(...)

A reference to a reference now stringifies as "REF(0x81485ec)" instead of "SCALAR(0x81485ec)" in order to be more consistent with the return value of ref().

pack/unpack D/F recycled

The undocumented pack/unpack template letters D/F have been recycled for better use: now they stand for long double (if supported by the platform) and NV (Perl internal floating point type). (They used to be aliases for d/f, but you never knew that.)

glob() now returns filenames in alphabetical order

The list of filenames from glob() (or <...>) is now by default sorted alphabetically to be csh-compliant (which is what happened before in most Unix platforms). (bsd_glob() does still sort platform natively, ASCII or EBCDIC, unless GLOB_ALPHASORT is specified.) [561]

Deprecations

- The semantics of bless(REF, REF) were unclear and until someone proves it to make some sense, it is forbidden.
- The obsolete chat2 library that should never have been allowed to escape the laboratory has been decommissioned.
- Using chdir("") or chdir(undef) instead of explicit chdir() is doubtful. A failure (think chdir(some_function()) can lead into unintended chdir() to the home directory, therefore this behaviour is deprecated.
- The builtin dump() function has probably outlived most of its usefulness. The core-dumping functionality will remain in future available as an explicit call to `CORE::dump()`, but in future releases the behaviour of an unqualified `dump()` call may change.
- The very dusty examples in the eg/ directory have been removed. Suggestions for new shiny examples welcome but the main issue is that the examples need to be documented, tested and (most importantly) maintained.
- The (bogus) escape sequences \8 and \9 now give an optional warning ("Unrecognized escape passed through"). There is no need to \-escape any `\w` character.
- The *glob{FILEHANDLE} is deprecated, use *glob{IO} instead.
- The `package;` syntax (`package` without an argument) has been deprecated. Its semantics were never that clear and its implementation even less so. If you have used that feature to disallow all but fully qualified variables, `use strict;` instead.
- The unimplemented POSIX regex features [[.cc.]] and [[=c=]] are still recognised but now cause fatal errors. The previous behaviour of ignoring them by default and warning if requested was unacceptable since it, in a way, falsely promised that the features could be used.
- In future releases, non-PerlIO aware XS modules may become completely unsupported. Since PerlIO is a drop-in replacement for stdio at the source code level, this shouldn't be that drastic a change.
- Previous versions of perl and some readings of some sections of Camel III implied that the `:raw` "discipline" was the inverse of `:crlf`. Turning off "clrfness" is no longer enough to make a stream truly binary. So the PerlIO `:raw` layer (or "discipline", to use the Camel book's older terminology) is now formally defined as being equivalent to binmode(FH) - which is in turn defined as doing whatever is necessary to pass each byte as-is without any translation. In particular binmode(FH) - and hence `:raw` - will now turn off both CRLF and UTF-8 translation and remove other layers (e.g. :encoding()) which would modify byte stream.
- The current user-visible implementation of pseudo-hashes (the weird use of the first array element) is deprecated starting from Perl 5.8.0 and will be removed in Perl 5.10.0, and the feature will be implemented differently. Not only is the current interface rather ugly, but the current implementation slows down normal array and hash use quite noticeably. The `fields`

pragma interface will remain available. The *restricted hashes* interface is expected to be the replacement interface (see *Hash::Util*). If your existing programs depends on the underlying implementation, consider using *Class::PseudoHash* from CPAN.

- The syntaxes `@a->[...]` and `%h->{...}` have now been deprecated.
- After years of trying, suidperl is considered to be too complex to ever be considered truly secure. The suidperl functionality is likely to be removed in a future release.
- The 5.005 threads model (module `Thread`) is deprecated and expected to be removed in Perl 5.10. Multithreaded code should be migrated to the new ithreads model (see *threads*, *threads::shared* and *perlthrtut*).
- The long deprecated uppercase aliases for the string comparison operators (EQ, NE, LT, LE, GE, GT) have now been removed.
- The tr///C and tr///U features have been removed and will not return; the interface was a mistake. Sorry about that. For similar functionality, see pack('U0', ...) and pack('C0', ...). [561]
- Earlier Perls treated "sub foo (@bar)" as equivalent to "sub foo (@)". The prototypes are now checked better at compile-time for invalid syntax. An optional warning is generated ("Illegal character in prototype...") but this may be upgraded to a fatal error in a future release.
- The `exec LIST` and `system LIST` operations now produce warnings on tainted data and in some future release they will produce fatal errors.
- The existing behaviour when localising tied arrays and hashes is wrong, and will be changed in a future release, so do not rely on the existing behaviour. See *Localising Tied Arrays and Hashes Is Broken*.

Core Enhancements

Unicode Overhaul

Unicode in general should be now much more usable than in Perl 5.6.0 (or even in 5.6.1). Unicode can be used in hash keys, Unicode in regular expressions should work now, Unicode in tr/// should work now, Unicode in I/O should work now. See *perluniintro* for introduction and *perlunicode* for details.

- The Unicode Character Database coming with Perl has been upgraded to Unicode 3.2.0. For more information, see http://www.unicode.org/ . [561+] (5.6.1 has UCD 3.0.1.)
- For developers interested in enhancing Perl's Unicode capabilities: almost all the UCD files are included with the Perl distribution in the *lib/unicore* subdirectory. The most notable omission, for space considerations, is the Unihan database.
- The properties \p{Blank} and \p{SpacePerl} have been added. "Blank" is like C isblank(), that is, it contains only "horizontal whitespace" (the space character is, the newline isn't), and the "SpacePerl" is the Unicode equivalent of `\s` (\p{Space} isn't, since that includes the vertical tabulator character, whereas `\s` doesn't.)

  See "New Unicode Properties" earlier in this document for additional information on changes with Unicode properties.

PerlIO is Now The Default

- IO is now by default done via PerlIO rather than system's "stdio". PerlIO allows "layers" to be "pushed" onto a file handle to alter the handle's behaviour. Layers can be specified at open time via 3-arg form of open:

  ```
  open($fh,'>:crlf :utf8', $path) || ...
  ```

  or on already opened handles via extended `binmode`:

```
binmode($fh,':encoding(iso-8859-7)');
```

The built-in layers are: unix (low level read/write), stdio (as in previous Perls), perlio (re-implementation of stdio buffering in a portable manner), crlf (does CRLF <=> "\n" translation as on Win32, but available on any platform). A mmap layer may be available if platform supports it (mostly Unixes).

Layers to be applied by default may be specified via the 'open' pragma.

See *Installation and Configuration Improvements* for the effects of PerlIO on your architecture name.

- If your platform supports fork(), you can use the list form of `open` for pipes. For example:

```
 open KID_PS, "-|", "ps", "aux" or die $!;
```

forks the ps(1) command (without spawning a shell, as there are more than three arguments to open()), and reads its standard output via the `KID_PS` filehandle. See *perlipc*.

- File handles can be marked as accepting Perl's internal encoding of Unicode (UTF-8 or UTF-EBCDIC depending on platform) by a pseudo layer ":utf8" :

```
open($fh,">:utf8","Uni.txt");
```

Note for EBCDIC users: the pseudo layer ":utf8" is erroneously named for you since it's not UTF-8 what you will be getting but instead UTF-EBCDIC. See *perlunicode*, *utf8*, and http://www.unicode.org/unicode/reports/tr16/ for more information. In future releases this naming may change. See *perluniintro* for more information about UTF-8.

- If your environment variables (LC_ALL, LC_CTYPE, LANG) look like you want to use UTF-8 (any of the variables match `/utf-?8/i`), your STDIN, STDOUT, STDERR handles and the default open layer (see *open*) are marked as UTF-8. (This feature, like other new features that combine Unicode and I/O, work only if you are using PerlIO, but that's the default.)

  Note that after this Perl really does assume that everything is UTF-8: for example if some input handle is not, Perl will probably very soon complain about the input data like this "Malformed UTF-8 ..." since any old eight-bit data is not legal UTF-8.

  Note for code authors: if you want to enable your users to use UTF-8 as their default encoding but in your code still have eight-bit I/O streams (such as images or zip files), you need to explicitly open() or binmode() with `:bytes` (see *"open" in perlfunc* and *"binmode" in perlfunc*), or you can just use `binmode(FH)` (nice for pre-5.8.0 backward compatibility).

- File handles can translate character encodings from/to Perl's internal Unicode form on read/write via the ":encoding()" layer.

- File handles can be opened to "in memory" files held in Perl scalars via:

```
open($fh,'>', \$variable) || ...
```

- Anonymous temporary files are available without need to 'use FileHandle' or other module via

```
open($fh,"+>", undef) || ...
```

That is a literal undef, not an undefined value.

ithreads

The new interpreter threads ("ithreads" for short) implementation of multithreading, by Arthur Bergman, replaces the old "5.005 threads" implementation. In the ithreads model any data sharing between threads must be explicit, as opposed to the model where data sharing was implicit. See *threads* and *threads::shared*, and *perlthrtut*.

As a part of the ithreads implementation Perl will also use any necessary and detectable reentrant libc

interfaces.

Restricted Hashes

A restricted hash is restricted to a certain set of keys, no keys outside the set can be added. Also individual keys can be restricted so that the key cannot be deleted and the value cannot be changed. No new syntax is involved: the Hash::Util module is the interface.

Safe Signals

Perl used to be fragile in that signals arriving at inopportune moments could corrupt Perl's internal state. Now Perl postpones handling of signals until it's safe (between opcodes).

This change may have surprising side effects because signals no longer interrupt Perl instantly. Perl will now first finish whatever it was doing, like finishing an internal operation (like sort()) or an external operation (like an I/O operation), and only then look at any arrived signals (and before starting the next operation). No more corrupt internal state since the current operation is always finished first, but the signal may take more time to get heard. Note that breaking out from potentially blocking operations should still work, though.

Understanding of Numbers

In general a lot of fixing has happened in the area of Perl's understanding of numbers, both integer and floating point. Since in many systems the standard number parsing functions like `strtoul()` and `atof()` seem to have bugs, Perl tries to work around their deficiencies. This results hopefully in more accurate numbers.

Perl now tries internally to use integer values in numeric conversions and basic arithmetics (+ - * /) if the arguments are integers, and tries also to keep the results stored internally as integers. This change leads to often slightly faster and always less lossy arithmetics. (Previously Perl always preferred floating point numbers in its math.)

Arrays now always interpolate into double-quoted strings [561]

In double-quoted strings, arrays now interpolate, no matter what. The behavior in earlier versions of perl 5 was that arrays would interpolate into strings if the array had been mentioned before the string was compiled, and otherwise Perl would raise a fatal compile-time error. In versions 5.000 through 5.003, the error was

```
Literal @example now requires backslash
```

In versions 5.004_01 through 5.6.0, the error was

```
In string, @example now must be written as \@example
```

The idea here was to get people into the habit of writing `"fred\@example.com"` when they wanted a literal @ sign, just as they have always written `"Give me back my \$5"` when they wanted a literal $ sign.

Starting with 5.6.1, when Perl now sees an @ sign in a double-quoted string, it *always* attempts to interpolate an array, regardless of whether or not the array has been used or declared already. The fatal error has been downgraded to an optional warning:

```
Possible unintended interpolation of @example in string
```

This warns you that `"fred@example.com"` is going to turn into `fred.com` if you don't backslash the @. See http://perl.plover.com/at-error.html for more details about the history here.

Miscellaneous Changes

- AUTOLOAD is now lvaluable, meaning that you can add the :lvalue attribute to AUTOLOAD subroutines and you can assign to the AUTOLOAD return value.

- The $Config{byteorder} (and corresponding BYTEORDER in config.h) was previously wrong in platforms if sizeof(long) was 4, but sizeof(IV) was 8. The byteorder was only sizeof(long) bytes long (1234 or 4321), but now it is correctly sizeof(IV) bytes long, (12345678 or 87654321). (This problem didn't affect Windows platforms.)

  Also, $Config{byteorder} is now computed dynamically--this is more robust with "fat binaries" where an executable image contains binaries for more than one binary platform, and when cross-compiling.

- `perl -d:Module=arg,arg,arg` now works (previously one couldn't pass in multiple arguments.)

- `do` followed by a bareword now ensures that this bareword isn't a keyword (to avoid a bug where `do q(foo.pl)` tried to call a subroutine called q). This means that for example instead of `do format()` you must write `do &format()`.

- The builtin dump() now gives an optional warning `dump() better written as CORE::dump()`, meaning that by default `dump(...)` is resolved as the builtin dump() which dumps core and aborts, not as (possibly) user-defined `sub dump`. To call the latter, qualify the call as `&dump(...)`. (The whole dump() feature is to considered deprecated, and possibly removed/changed in future releases.)

- chomp() and chop() are now overridable. Note, however, that their prototype (as given by `prototype("CORE::chomp")`) is undefined, because it cannot be expressed and therefore one cannot really write replacements to override these builtins.

- END blocks are now run even if you exit/die in a BEGIN block. Internally, the execution of END blocks is now controlled by PL_exit_flags & PERL_EXIT_DESTRUCT_END. This enables the new behaviour for Perl embedders. This will default in 5.10. See *perlembed*.

- Formats now support zero-padded decimal fields.

- Although "you shouldn't do that", it was possible to write code that depends on Perl's hashed key order (Data::Dumper does this). The new algorithm "One-at-a-Time" produces a different hashed key order. More details are in *Performance Enhancements*.

- lstat(FILEHANDLE) now gives a warning because the operation makes no sense. In future releases this may become a fatal error.

- Spurious syntax errors generated in certain situations, when glob() caused File::Glob to be loaded for the first time, have been fixed. [561]

- Lvalue subroutines can now return `undef` in list context. However, the lvalue subroutine feature still remains experimental. [561+]

- A lost warning "Can't declare ... dereference in my" has been restored (Perl had it earlier but it became lost in later releases.)

- A new special regular expression variable has been introduced: `$^N`, which contains the most-recently closed group (submatch).

- `no Module;` does not produce an error even if Module does not have an unimport() method. This parallels the behavior of `use` vis-a-vis `import`. [561]

- The numerical comparison operators return `undef` if either operand is a NaN. Previously the behaviour was unspecified.

- `our` can now have an experimental optional attribute `unique` that affects how global variables are shared among multiple interpreters, see *"our" in perlfunc*.

- The following builtin functions are now overridable: each(), keys(), pop(), push(), shift(), splice(), unshift(). [561]

- `pack() / unpack()` can now group template letters with `()` and then apply repetition/count modifiers on the groups.
- `pack() / unpack()` can now process the Perl internal numeric types: IVs, UVs, NVs-- and also long doubles, if supported by the platform. The template letters are `j`, `J`, `F`, and `D`.
- `pack('U0a*', ...)` can now be used to force a string to UTF-8.
- my __PACKAGE__ $obj now works. [561]
- POSIX::sleep() now returns the number of *unslept* seconds (as the POSIX standard says), as opposed to CORE::sleep() which returns the number of slept seconds.
- printf() and sprintf() now support parameter reordering using the `%\d+\$` and `*\d+\$` syntaxes. For example

  ```
  printf "%2\$s %1\$s\n", "foo", "bar";
  ```

  will print "bar foo\n". This feature helps in writing internationalised software, and in general when the order of the parameters can vary.
- The (\&) prototype now works properly. [561]
- prototype(\[$@%&]) is now available to implicitly create references (useful for example if you want to emulate the tie() interface).
- A new command-line option, `-t` is available. It is the little brother of `-T`: instead of dying on taint violations, lexical warnings are given. **This is only meant as a temporary debugging aid while securing the code of old legacy applications. This is not a substitute for -T.**
- In other taint news, the `exec LIST` and `system LIST` have now been considered too risky (think `exec @ARGV`: it can start any program with any arguments), and now the said forms cause a warning under lexical warnings. You should carefully launder the arguments to guarantee their validity. In future releases of Perl the forms will become fatal errors so consider starting laundering now.
- Tied hash interfaces are now required to have the EXISTS and DELETE methods (either own or inherited).
- If tr/// is just counting characters, it doesn't attempt to modify its target.
- untie() will now call an UNTIE() hook if it exists. See *perltie* for details. [561]
- *"utime" in perlfunc* now supports `utime undef, undef, @files` to change the file timestamps to the current time.
- The rules for allowing underscores (underbars) in numeric constants have been relaxed and simplified: now you can have an underscore simply **between digits**.
- Rather than relying on C's argv[0] (which may not contain a full pathname) where possible $^X is now set by asking the operating system. (eg by reading */proc/self/exe* on Linux, */proc/curproc/file* on FreeBSD)
- A new variable, `${^TAINT}`, indicates whether taint mode is enabled.
- You can now override the readline() builtin, and this overrides also the <FILEHANDLE> angle bracket operator.
- The command-line options -s and -F are now recognized on the shebang (#!) line.
- Use of the `/c` match modifier without an accompanying `/g` modifier elicits a new warning: `Use of /c modifier is meaningless without /g`.

  Use of `/c` in substitutions, even with `/g`, elicits `Use of /c modifier is meaningless`

in s///.

Use of /g with split elicits Use of /g modifier is meaningless in split.

- Support for the CLONE special subroutine had been added. With ithreads, when a new thread is created, all Perl data is cloned, however non-Perl data cannot be cloned automatically. In CLONE you can do whatever you need to do, like for example handle the cloning of non-Perl data, if necessary. CLONE will be executed once for every package that has it defined or inherited. It will be called in the context of the new thread, so all modifications are made in the new area.

  See *perlmod*

Modules and Pragmata

New Modules and Pragmata

- Attribute::Handlers, originally by Damian Conway and now maintained by Arthur Bergman, allows a class to define attribute handlers.

```
    package MyPack;
    use Attribute::Handlers;
    sub Wolf :ATTR(SCALAR) { print "howl!\n" }

    # later, in some package using or inheriting from MyPack...

    my MyPack $Fluffy : Wolf; # the attribute handler Wolf will be
called
```

  Both variables and routines can have attribute handlers. Handlers can be specific to type (SCALAR, ARRAY, HASH, or CODE), or specific to the exact compilation phase (BEGIN, CHECK, INIT, or END). See *Attribute::Handlers*.

- B::Concise, by Stephen McCamant, is a new compiler backend for walking the Perl syntax tree, printing concise info about ops. The output is highly customisable. See *B::Concise*. [561+]

- The new bignum, bigint, and bigrat pragmas, by Tels, implement transparent bignum support (using the Math::BigInt, Math::BigFloat, and Math::BigRat backends).

- Class::ISA, by Sean Burke, is a module for reporting the search path for a class's ISA tree. See *Class::ISA*.

- Cwd now has a split personality: if possible, an XS extension is used, (this will hopefully be faster, more secure, and more robust) but if not possible, the familiar Perl implementation is used.

- Devel::PPPort, originally by Kenneth Albanowski and now maintained by Paul Marquess, has been added. It is primarily used by h2xs to enhance portability of XS modules between different versions of Perl. See *Devel::PPPort*.

- Digest, frontend module for calculating digests (checksums), from Gisle Aas, has been added. See *Digest*.

- Digest::MD5 for calculating MD5 digests (checksums) as defined in RFC 1321, from Gisle Aas, has been added. See *Digest::MD5*.

```
    use Digest::MD5 'md5_hex';

    $digest = md5_hex("Thirsty Camel");

    print $digest, "\n"; # 01d19d9d2045e005c3f1b80e8b164de1
```

NOTE: the `MD5` backward compatibility module is deliberately not included since its further use is discouraged.

See also *PerlIO::via::QuotedPrint*.

- `Encode`, originally by Nick Ing-Simmons and now maintained by Dan Kogai, provides a mechanism to translate between different character encodings. Support for Unicode, ISO-8859-1, and ASCII are compiled in to the module. Several other encodings (like the rest of the ISO-8859, CP*/Win*, Mac, KOI8-R, three variants EBCDIC, Chinese, Japanese, and Korean encodings) are included and can be loaded at runtime. (For space considerations, the largest Chinese encodings have been separated into their own CPAN module, Encode::HanExtra, which Encode will use if available). See *Encode*.

  Any encoding supported by Encode module is also available to the ":encoding()" layer if PerlIO is used.

- `Hash::Util` is the interface to the new *restricted hashes* feature. (Implemented by Jeffrey Friedl, Nick Ing-Simmons, and Michael Schwern.) See *Hash::Util*.

- `I18N::Langinfo` can be used to query locale information. See *I18N::Langinfo*.

- `I18N::LangTags`, by Sean Burke, has functions for dealing with RFC3066-style language tags. See *I18N::LangTags*.

- `ExtUtils::Constant`, by Nicholas Clark, is a new tool for extension writers for generating XS code to import C header constants. See *ExtUtils::Constant*.

- `Filter::Simple`, by Damian Conway, is an easy-to-use frontend to Filter::Util::Call. See *Filter::Simple*.

```
# in MyFilter.pm:

package MyFilter;

use Filter::Simple sub {
    while (my ($from, $to) = splice @_, 0, 2) {
            s/$from/$to/g;
    }
};

1;

# in user's code:

use MyFilter qr/red/ => 'green';

print "red\n";   # this code is filtered, will print "green\n"
print "bored\n"; # this code is filtered, will print "bogreen\n"

no MyFilter;

print "red\n";   # this code is not filtered, will print "red\n"
```

- `File::Temp`, by Tim Jenness, allows one to create temporary files and directories in an easy, portable, and secure way. See *File::Temp*. [561+]

- `Filter::Util::Call`, by Paul Marquess, provides you with the framework to write *source filters* in Perl. For most uses, the frontend Filter::Simple is to be preferred. See *Filter::Util::Call*.

- `if`, by Ilya Zakharevich, is a new pragma for conditional inclusion of modules.

- *libnet*, by Graham Barr, is a collection of perl5 modules related to network programming. See *Net::FTP*, *Net::NNTP*, *Net::Ping* (not part of libnet, but related), *Net::POP3*, *Net::SMTP*, and *Net::Time*.

  Perl installation leaves libnet unconfigured; use *libnetcfg* to configure it.

- `List::Util`, by Graham Barr, is a selection of general-utility list subroutines, such as sum(), min(), first(), and shuffle(). See *List::Util*.

- `Locale::Constants`, `Locale::Country`, `Locale::Currency` `Locale::Language`, and *Locale::Script*, by Neil Bowers, have been added. They provide the codes for various locale standards, such as "fr" for France, "usd" for US Dollar, and "ja" for Japanese.

  ```
      use Locale::Country;

      $country = code2country('jp');               # $country gets
  'Japan'
      $code    = country2code('Norway');           # $code gets 'no'
  ```

  See *Locale::Constants*, *Locale::Country*, *Locale::Currency*, and *Locale::Language*.

- `Locale::Maketext`, by Sean Burke, is a localization framework. See *Locale::Maketext*, and *Locale::Maketext::TPJ13*. The latter is an article about software localization, originally published in The Perl Journal #13, and republished here with kind permission.

- `Math::BigRat` for big rational numbers, to accompany Math::BigInt and Math::BigFloat, from Tels. See *Math::BigRat*.

- `Memoize` can make your functions faster by trading space for time, from Mark-Jason Dominus. See *Memoize*.

- `MIME::Base64`, by Gisle Aas, allows you to encode data in base64, as defined in RFC 2045 - *MIME (Multipurpose Internet Mail Extensions)*.

  ```
      use MIME::Base64;

      $encoded = encode_base64('Aladdin:open sesame');
      $decoded = decode_base64($encoded);

      print $encoded, "\n"; # "QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
  ```

  See *MIME::Base64*.

- `MIME::QuotedPrint`, by Gisle Aas, allows you to encode data in quoted-printable encoding, as defined in RFC 2045 - *MIME (Multipurpose Internet Mail Extensions)*.

  ```
      use MIME::QuotedPrint;

      $encoded = encode_qp("\xDE\xAD\xBE\xEF");
      $decoded = decode_qp($encoded);

      print $encoded, "\n"; # "=DE=AD=BE=EF\n"
      print $decoded, "\n"; # "\xDE\xAD\xBE\xEF\n"
  ```

  See also *PerlIO::via::QuotedPrint*.

- `NEXT`, by Damian Conway, is a pseudo-class for method redispatch. See *NEXT*.

- `open` is a new pragma for setting the default I/O layers for open().

- `PerlIO::scalar`, by Nick Ing-Simmons, provides the implementation of IO to "in memory" Perl scalars as discussed above. It also serves as an example of a loadable PerlIO layer.

Other future possibilities include PerlIO::Array and PerlIO::Code. See *PerlIO::scalar*.

- `PerlIO::via`, by Nick Ing-Simmons, acts as a PerlIO layer and wraps PerlIO layer functionality provided by a class (typically implemented in Perl code).

- `PerlIO::via::QuotedPrint`, by Elizabeth Mattijsen, is an example of a `PerlIO::via` class:

```
use PerlIO::via::QuotedPrint;
open($fh,">:via(QuotedPrint)",$path);
```

  This will automatically convert everything output to `$fh` to Quoted-Printable. See *PerlIO::via* and *PerlIO::via::QuotedPrint*.

- `Pod::ParseLink`, by Russ Allbery, has been added, to parse L<> links in pods as described in the new perlpodspec.

- `Pod::Text::Overstrike`, by Joe Smith, has been added. It converts POD data to formatted overstrike text. See *Pod::Text::Overstrike*. [561+]

- `Scalar::Util` is a selection of general-utility scalar subroutines, such as blessed(), reftype(), and tainted(). See *Scalar::Util*.

- `sort` is a new pragma for controlling the behaviour of sort().

- `Storable` gives persistence to Perl data structures by allowing the storage and retrieval of Perl data to and from files in a fast and compact binary format. Because in effect Storable does serialisation of Perl data structures, with it you can also clone deep, hierarchical datastructures. Storable was originally created by Raphael Manfredi, but it is now maintained by Abhijit Menon-Sen. Storable has been enhanced to understand the two new hash features, Unicode keys and restricted hashes. See *Storable*.

- `Switch`, by Damian Conway, has been added. Just by saying

```
use Switch;
```

  you have `switch` and `case` available in Perl.

```
use Switch;

switch ($val) {

case 1  { print "number 1" }
case "a" { print "string a" }
case [1..10,42] { print "number in list" }
case (@array) { print "number in list" }
case /\w+/ { print "pattern" }
case qr/\w+/ { print "pattern" }
case (%hash) { print "entry in hash" }
case (\%hash) { print "entry in hash" }
case (\&sub) { print "arg to subroutine" }
else  { print "previous case not true" }
  }
```

  See *Switch*.

- `Test::More`, by Michael Schwern, is yet another framework for writing test scripts, more extensive than Test::Simple. See *Test::More*.

- `Test::Simple`, by Michael Schwern, has basic utilities for writing tests. See *Test::Simple*.

- `Text::Balanced`, by Damian Conway, has been added, for extracting delimited text

sequences from strings.

```
    use Text::Balanced 'extract_delimited';

    ($a, $b) = extract_delimited("'never say never', he never said",
"'", '');
```

$a will be "'never say never'", $b will be ', he never said'.

In addition to extract_delimited(), there are also extract_bracketed(), extract_quotelike(), extract_codeblock(), extract_variable(), extract_tagged(), extract_multiple(), gen_delimited_pat(), and gen_extract_tagged(). With these, you can implement rather advanced parsing algorithms. See *Text::Balanced*.

- `threads`, by Arthur Bergman, is an interface to interpreter threads. Interpreter threads (ithreads) is the new thread model introduced in Perl 5.6 but only available as an internal interface for extension writers (and for Win32 Perl for `fork()` emulation). See *threads*, *threads::shared*, and *perlthrtut*.
- `threads::shared`, by Arthur Bergman, allows data sharing for interpreter threads. See *threads::shared*.
- `Tie::File`, by Mark-Jason Dominus, associates a Perl array with the lines of a file. See *Tie::File*.
- `Tie::Memoize`, by Ilya Zakharevich, provides on-demand loaded hashes. See *Tie::Memoize*.
- `Tie::RefHash::Nestable`, by Edward Avis, allows storing hash references (unlike the standard Tie::RefHash) The module is contained within Tie::RefHash. See *Tie::RefHash*.
- `Time::HiRes`, by Douglas E. Wegscheid, provides high resolution timing (ualarm, usleep, and gettimeofday). See *Time::HiRes*.
- `Unicode::UCD` offers a querying interface to the Unicode Character Database. See *Unicode::UCD*.
- `Unicode::Collate`, by SADAHIRO Tomoyuki, implements the UCA (Unicode Collation Algorithm) for sorting Unicode strings. See *Unicode::Collate*.
- `Unicode::Normalize`, by SADAHIRO Tomoyuki, implements the various Unicode normalization forms. See *Unicode::Normalize*.
- `XS::APItest`, by Tim Jenness, is a test extension that exercises XS APIs. Currently only `printf()` is tested: how to output various basic data types from XS.
- `XS::Typemap`, by Tim Jenness, is a test extension that exercises XS typemaps. Nothing gets installed, but the code is worth studying for extension writers.

Updated And Improved Modules and Pragmata

- The following independently supported modules have been updated to the newest versions from CPAN: CGI, CPAN, DB_File, File::Spec, File::Temp, Getopt::Long, Math::BigFloat, Math::BigInt, the podlators bundle (Pod::Man, Pod::Text), Pod::LaTeX [561+], Pod::Parser, Storable, Term::ANSIColor, Test, Text-Tabs+Wrap.
- attributes::reftype() now works on tied arguments.
- AutoLoader can now be disabled with `no AutoLoader;`.
- B::Deparse has been significantly enhanced by Robin Houston. It can now deparse almost all of the standard test suite (so that the tests still succeed). There is a make target "test.deparse" for trying this out.

- Carp now has better interface documentation, and the @CARP_NOT interface has been added to get optional control over where errors are reported independently of @ISA, by Ben Tilly.
- Class::Struct can now define the classes in compile time.
- Class::Struct now assigns the array/hash element if the accessor is called with an array/hash element as the **sole** argument.
- The return value of Cwd::fastcwd() is now tainted.
- Data::Dumper now has an option to sort hashes.
- Data::Dumper now has an option to dump code references using B::Deparse.
- DB_File now supports newer Berkeley DB versions, among other improvements.
- Devel::Peek now has an interface for the Perl memory statistics (this works only if you are using perl's malloc, and if you have compiled with debugging).
- The English module can now be used without the infamous performance hit by saying

  ```
  use English '-no_match_vars';
  ```

  (Assuming, of course, that you don't need the troublesome variables $`, $&, or $'.) Also, introduced `@LAST_MATCH_START` and `@LAST_MATCH_END` English aliases for `@-` and `@+`.
- ExtUtils::MakeMaker has been significantly cleaned up and fixed. The enhanced version has also been backported to earlier releases of Perl and submitted to CPAN so that the earlier releases can enjoy the fixes.
- The arguments of WriteMakefile() in Makefile.PL are now checked for sanity much more carefully than before. This may cause new warnings when modules are being installed. See *ExtUtils::MakeMaker* for more details.
- ExtUtils::MakeMaker now uses File::Spec internally, which hopefully leads to better portability.
- Fcntl, Socket, and Sys::Syslog have been rewritten by Nicholas Clark to use the new-style constant dispatch section (see *ExtUtils::Constant*). This means that they will be more robust and hopefully faster.
- File::Find now chdir()s correctly when chasing symbolic links. [561]
- File::Find now has pre- and post-processing callbacks. It also correctly changes directories when chasing symbolic links. Callbacks (naughtily) exiting with "next;" instead of "return;" now work.
- File::Find is now (again) reentrant. It also has been made more portable.
- The warnings issued by File::Find now belong to their own category. You can enable/disable them with `use/no warnings 'File::Find';`.
- File::Glob::glob() has been renamed to File::Glob::bsd_glob() because the name clashes with the builtin glob(). The older name is still available for compatibility, but is deprecated. [561]
- File::Glob now supports `GLOB_LIMIT` constant to limit the size of the returned list of filenames.
- IPC::Open3 now allows the use of numeric file descriptors.
- IO::Socket now has an atmark() method, which returns true if the socket is positioned at the out-of-band mark. The method is also exportable as a sockatmark() function.
- IO::Socket::INET failed to open the specified port if the service name was not known. It now

correctly uses the supplied port number as is. [561]

- IO::Socket::INET has support for the ReusePort option (if your platform supports it). The Reuse option now has an alias, ReuseAddr. For clarity, you may want to prefer ReuseAddr.
- IO::Socket::INET now supports a value of zero for `LocalPort` (usually meaning that the operating system will make one up.)
- 'use lib' now works identically to @INC. Removing directories with 'no lib' now works.
- Math::BigFloat and Math::BigInt have undergone a full rewrite by Tels. They are now magnitudes faster, and they support various bignum libraries such as GMP and PARI as their backends.
- Math::Complex handles inf, NaN etc., better.
- Net::Ping has been considerably enhanced by Rob Brown: multihoming is now supported, Win32 functionality is better, there is now time measuring functionality (optionally high-resolution using Time::HiRes), and there is now "external" protocol which uses Net::Ping::External module which runs your external ping utility and parses the output. A version of Net::Ping::External is available in CPAN.

  Note that some of the Net::Ping tests are disabled when running under the Perl distribution since one cannot assume one or more of the following: enabled echo port at localhost, full Internet connectivity, or sympathetic firewalls. You can set the environment variable PERL_TEST_Net_Ping to "1" (one) before running the Perl test suite to enable all the Net::Ping tests.
- POSIX::sigaction() is now much more flexible and robust. You can now install coderef handlers, 'DEFAULT', and 'IGNORE' handlers, installing new handlers was not atomic.
- In Safe, `%INC` is now localised in a Safe compartment so that use/require work.
- In SDBM_File on dosish platforms, some keys went missing because of lack of support for files with "holes". A workaround for the problem has been added.
- In Search::Dict one can now have a pre-processing hook for the lines being searched.
- The Shell module now has an OO interface.
- In Sys::Syslog there is now a failover mechanism that will go through alternative connection mechanisms until the message is successfully logged.
- The Test module has been significantly enhanced.
- Time::Local::timelocal() does not handle fractional seconds anymore. The rationale is that neither does localtime(), and timelocal() and localtime() are supposed to be inverses of each other.
- The vars pragma now supports declaring fully qualified variables. (Something that `our()` does not and will not support.)
- The `utf8::` name space (as in the pragma) provides various Perl-callable functions to provide low level access to Perl's internal Unicode representation. At the moment only length() has been implemented.

Utility Changes

- Emacs perl mode (emacs/cperl-mode.el) has been updated to version 4.31.
- *emacs/e2ctags.pl* is now much faster.
- `enc2xs` is a tool for people adding their own encodings to the Encode module.

- `h2ph` now supports C trigraphs.
- `h2xs` now produces a template README.
- `h2xs` now uses `Devel::PPPort` for better portability between different versions of Perl.
- `h2xs` uses the new *ExtUtils::Constant* module which will affect newly created extensions that define constants. Since the new code is more correct (if you have two constants where the first one is a prefix of the second one, the first constant **never** got defined), less lossy (it uses integers for integer constant, as opposed to the old code that used floating point numbers even for integer constants), and slightly faster, you might want to consider regenerating your extension code (the new scheme makes regenerating easy). *h2xs* now also supports C trigraphs.
- `libnetcfg` has been added to configure libnet.
- `perlbug` is now much more robust. It also sends the bug report to perl.org, not perl.com.
- `perlcc` has been rewritten and its user interface (that is, command line) is much more like that of the Unix C compiler, cc. (The perlbc tools has been removed. Use `perlcc -B` instead.) **Note that perlcc is still considered very experimental and unsupported.** [561]
- `perlivp` is a new Installation Verification Procedure utility for running any time after installing Perl.
- `piconv` is an implementation of the character conversion utility `iconv`, demonstrating the new Encode module.
- `pod2html` now allows specifying a cache directory.
- `pod2html` now produces XHTML 1.0.
- `pod2html` now understands POD written using different line endings (PC-like CRLF versus Unix-like LF versus MacClassic-like CR).
- `s2p` has been completely rewritten in Perl. (It is in fact a full implementation of sed in Perl: you can use the sed functionality by using the `psed` utility.)
- `xsubpp` now understands POD documentation embedded in the *.xs files. [561]
- `xsubpp` now supports the OUT keyword.

New Documentation

- perl56delta details the changes between the 5.005 release and the 5.6.0 release.
- perlclib documents the internal replacements for standard C library functions. (Interesting only for extension writers and Perl core hackers.) [561+]
- perldebtut is a Perl debugging tutorial. [561+]
- perlebcdic contains considerations for running Perl on EBCDIC platforms. [561+]
- perlintro is a gentle introduction to Perl.
- perliol documents the internals of PerlIO with layers.
- perlmodstyle is a style guide for writing modules.
- perlnewmod tells about writing and submitting a new module. [561+]
- perlpacktut is a pack() tutorial.
- perlpod has been rewritten to be clearer and to record the best practices gathered over the years.

- perlpodspec is a more formal specification of the pod format, mainly of interest for writers of pod applications, not to people writing in pod.
- perlretut is a regular expression tutorial. [561+]
- perlrequick is a regular expressions quick-start guide. Yes, much quicker than perlretut. [561]
- perltodo has been updated.
- perltootc has been renamed as perltooc (to not to conflict with perltoot in filesystems restricted to "8.3" names).
- perluniintro is an introduction to using Unicode in Perl. (perlunicode is more of a detailed reference and background information)
- perlutil explains the command line utilities packaged with the Perl distribution. [561+]

The following platform-specific documents are available before the installation as README.*platform*, and after the installation as perl*platform*:

```
perlaix perlamiga perlapollo perlbeos perlbs2000
perlce perlcygwin perldgux perldos perlepoc perlfreebsd perlhpux
perlhurd perlirix perlmachten perlmacos perlmint perlmpeix
perlnetware perlos2 perlos390 perlplan9 perlqnx perlsolaris
perltru64 perluts perlvmesa perlvms perlvos perlwin32
```

These documents usually detail one or more of the following subjects: configuring, building, testing, installing, and sometimes also using Perl on the said platform.

Eastern Asian Perl users are now welcomed in their own languages: README.jp (Japanese), README.ko (Korean), README.cn (simplified Chinese) and README.tw (traditional Chinese), which are written in normal pod but encoded in EUC-JP, EUC-KR, EUC-CN and Big5. These will get installed as

```
perljp perlko perlcn perltw
```

- The documentation for the POSIX-BC platform is called "BS2000", to avoid confusion with the Perl POSIX module.
- The documentation for the WinCE platform is called perlce (README.ce in the source code kit), to avoid confusion with the perlwin32 documentation on 8.3-restricted filesystems.

Performance Enhancements

- map() could get pathologically slow when the result list it generates is larger than the source list. The performance has been improved for common scenarios. [561]
- sort() is also fully reentrant, in the sense that the sort function can itself call sort(). This did not work reliably in previous releases. [561]
- sort() has been changed to use primarily mergesort internally as opposed to the earlier quicksort. For very small lists this may result in slightly slower sorting times, but in general the speedup should be at least 20%. Additional bonuses are that the worst case behaviour of sort() is now better (in computer science terms it now runs in time O(N log N), as opposed to quicksort's Theta(N**2) worst-case run time behaviour), and that sort() is now stable (meaning that elements with identical keys will stay ordered as they were before the sort). See the `sort` pragma for information.

  The story in more detail: suppose you want to serve yourself a little slice of Pi.

  ```
  @digits = ( 3,1,4,1,5,9 );
  ```

A numerical sort of the digits will yield (1,1,3,4,5,9), as expected. Which 1 comes first is hard to know, since one 1 looks pretty much like any other. You can regard this as totally trivial, or somewhat profound. However, if you just want to sort the even digits ahead of the odd ones, then what will

```
sort { ($a % 2) <=> ($b % 2) } @digits;
```

yield? The only even digit, 4, will come first. But how about the odd numbers, which all compare equal? With the quicksort algorithm used to implement Perl 5.6 and earlier, the order of ties is left up to the sort. So, as you add more and more digits of Pi, the order in which the sorted even and odd digits appear will change. and, for sufficiently large slices of Pi, the quicksort algorithm in Perl 5.8 won't return the same results even if reinvoked with the same input. The justification for this rests with quicksort's worst case behavior. If you run

```
sort { $a <=> $b } ( 1 .. $N , 1 .. $N );
```

(something you might approximate if you wanted to merge two sorted arrays using sort), doubling $N doesn't just double the quicksort time, it *quadruples* it. Quicksort has a worst case run time that can grow like N**2, so-called *quadratic* behaviour, and it can happen on patterns that may well arise in normal use. You won't notice this for small arrays, but you *will* notice it with larger arrays, and you may not live long enough for the sort to complete on arrays of a million elements. So the 5.8 quicksort scrambles large arrays before sorting them, as a statistical defence against quadratic behaviour. But that means if you sort the same large array twice, ties may be broken in different ways.

Because of the unpredictability of tie-breaking order, and the quadratic worst-case behaviour, quicksort was *almost* replaced completely with a stable mergesort. *Stable* means that ties are broken to preserve the original order of appearance in the input array. So

```
sort { ($a % 2) <=> ($b % 2) } (3,1,4,1,5,9);
```

will yield (4,3,1,1,5,9), guaranteed. The even and odd numbers appear in the output in the same order they appeared in the input. Mergesort has worst case O(N log N) behaviour, the best value attainable. And, ironically, this mergesort does particularly well where quicksort goes quadratic: mergesort sorts (1..$N, 1..$N) in O(N) time. But quicksort was rescued at the last moment because it is faster than mergesort on certain inputs and platforms. For example, if you really *don't* care about the order of even and odd digits, quicksort will run in O(N) time; it's very good at sorting many repetitions of a small number of distinct elements. The quicksort divide and conquer strategy works well on platforms with relatively small, very fast, caches. Eventually, the problem gets whittled down to one that fits in the cache, from which point it benefits from the increased memory speed.

Quicksort was rescued by implementing a sort pragma to control aspects of the sort. The **stable** subpragma forces stable behaviour, regardless of algorithm. The **_quicksort** and **_mergesort** subpragmas are heavy-handed ways to select the underlying implementation. The leading _ is a reminder that these subpragmas may not survive beyond 5.8. More appropriate mechanisms for selecting the implementation exist, but they wouldn't have arrived in time to save quicksort.

- Hashes now use Bob Jenkins "One-at-a-Time" hashing key algorithm (http://burtleburtle.net/bob/hash/doobs.html). This algorithm is reasonably fast while producing a much better spread of values than the old hashing algorithm (originally by Chris Torek, later tweaked by Ilya Zakharevich). Hash values output from the algorithm on a hash of all 3-char printable ASCII keys comes much closer to passing the DIEHARD random number generation tests. According to perlbench, this change has not affected the overall speed of Perl.

- unshift() should now be noticeably faster.

Installation and Configuration Improvements

Generic Improvements

- INSTALL now explains how you can configure Perl to use 64-bit integers even on non-64-bit platforms.
- Policy.sh policy change: if you are reusing a Policy.sh file (see INSTALL) and you use Configure -Dprefix=/foo/bar and in the old Policy $prefix eq $siteprefix and $prefix eq $vendorprefix, all of them will now be changed to the new prefix, /foo/bar. (Previously only $prefix changed.) If you do not like this new behaviour, specify prefix, siteprefix, and vendorprefix explicitly.
- A new optional location for Perl libraries, otherlibdirs, is available. It can be used for example for vendor add-ons without disturbing Perl's own library directories.
- In many platforms, the vendor-supplied 'cc' is too stripped-down to build Perl (basically, 'cc' doesn't do ANSI C). If this seems to be the case and 'cc' does not seem to be the GNU C compiler 'gcc', an automatic attempt is made to find and use 'gcc' instead.
- gcc needs to closely track the operating system release to avoid build problems. If Configure finds that gcc was built for a different operating system release than is running, it now gives a clearly visible warning that there may be trouble ahead.
- Since Perl 5.8 is not binary-compatible with previous releases of Perl, Configure no longer suggests including the 5.005 modules in @INC.
- Configure `-S` can now run non-interactively. [561]
- Configure support for pdp11-style memory models has been removed due to obsolescence. [561]
- configure.gnu now works with options with whitespace in them.
- installperl now outputs everything to STDERR.
- Because PerlIO is now the default on most platforms, "-perlio" doesn't get appended to the $Config{archname} (also known as $^O) anymore. Instead, if you explicitly choose not to use perlio (Configure command line option -Uuseperlio), you will get "-stdio" appended.
- Another change related to the architecture name is that "-64all" (-Duse64bitall, or "maximally 64-bit") is appended only if your pointers are 64 bits wide. (To be exact, the use64bitall is ignored.)
- In AFS installations, one can configure the root of the AFS to be somewhere else than the default */afs* by using the Configure parameter `-Dafsroot=/some/where/else`.
- APPLLIB_EXP, a lesser-known configuration-time definition, has been documented. It can be used to prepend site-specific directories to Perl's default search path (@INC); see INSTALL for information.
- The version of Berkeley DB used when the Perl (and, presumably, the DB_File extension) was built is now available as `@Config{qw(db_version_major db_version_minor db_version_patch)}` from Perl and as `DB_VERSION_MAJOR_CFG DB_VERSION_MINOR_CFG DB_VERSION_PATCH_CFG` from C.
- Building Berkeley DB3 for compatibility modes for DB, NDBM, and ODBM has been documented in INSTALL.
- If you have CPAN access (either network or a local copy such as a CD-ROM) you can during specify extra modules to Configure to build and install with Perl using the -Dextras=... option. See INSTALL for more details.

- In addition to config.over, a new override file, config.arch, is available. This file is supposed to be used by hints file writers for architecture-wide changes (as opposed to config.over which is for site-wide changes).

- If your file system supports symbolic links, you can build Perl outside of the source directory by

  ```
  mkdir perl/build/directory
  cd perl/build/directory
  sh /path/to/perl/source/Configure -Dmksymlinks ...
  ```

  This will create in perl/build/directory a tree of symbolic links pointing to files in /path/to/perl/source. The original files are left unaffected. After Configure has finished, you can just say

  ```
  make all test
  ```

  and Perl will be built and tested, all in perl/build/directory. [561]

- For Perl developers, several new make targets for profiling and debugging have been added; see *perlhack*.

  - Use of the *gprof* tool to profile Perl has been documented in *perlhack*. There is a make target called "perl.gprof" for generating a gprofiled Perl executable.

  - If you have GCC 3, there is a make target called "perl.gcov" for creating a gcoved Perl executable for coverage analysis. See *perlhack*.

  - If you are on IRIX or Tru64 platforms, new profiling/debugging options have been added; see *perlhack* for more information about pixie and Third Degree.

- Guidelines of how to construct minimal Perl installations have been added to INSTALL.

- The Thread extension is now not built at all under ithreads (`Configure -Duseithreads`) because it wouldn't work anyway (the Thread extension requires being Configured with `-Duse5005threads`).

  Note that the 5.005 threads are unsupported and deprecated: if you have code written for the old threads you should migrate it to the new ithreads model.

- The Gconvert macro ($Config{d_Gconvert}) used by perl for stringifying floating-point numbers is now more picky about using sprintf %.*g rules for the conversion. Some platforms that used to use gcvt may now resort to the slower sprintf.

- The obsolete method of making a special (e.g., debugging) flavor of perl by saying

  ```
  make LIBPERL=libperld.a
  ```

  has been removed. Use -DDEBUGGING instead.

New Or Improved Platforms

For the list of platforms known to support Perl, see *"Supported Platforms" in perlport*.

- AIX dynamic loading should be now better supported.

- AIX should now work better with gcc, threads, and 64-bitness. Also the long doubles support in AIX should be better now. See *perlaix*.

- AtheOS (http://www.atheos.cx/) is a new platform.

- BeOS has been reclaimed.

- The DG/UX platform now supports 5.005-style threads. See *perldgux*.

- The DYNIX/ptx platform (also known as dynixptx) is supported at or near osvers 4.5.2.
- EBCDIC platforms (z/OS (also known as OS/390), POSIX-BC, and VM/ESA) have been regained. Many test suite tests still fail and the co-existence of Unicode and EBCDIC isn't quite settled, but the situation is much better than with Perl 5.6. See *perlos390*, *perlbs2000* (for POSIX-BC), and perlvmesa for more information. (**Note:** support for VM/ESA was removed in Perl v5.18.0. The relevant information was in *README.vmesa*)
- Building perl with -Duseithreads or -Duse5005threads now works under HP-UX 10.20 (previously it only worked under 10.30 or later). You will need a thread library package installed. See README.hpux. [561]
- Mac OS Classic is now supported in the mainstream source package (MacPerl has of course been available since perl 5.004 but now the source code bases of standard Perl and MacPerl have been synchronised) [561]
- Mac OS X (or Darwin) should now be able to build Perl even on HFS+ filesystems. (The case-insensitivity used to confuse the Perl build process.)
- NCR MP-RAS is now supported. [561]
- All the NetBSD specific patches (except for the installation specific ones) have been merged back to the main distribution.
- NetWare from Novell is now supported. See *perlnetware*.
- NonStop-UX is now supported. [561]
- NEC SUPER-UX is now supported.
- All the OpenBSD specific patches (except for the installation specific ones) have been merged back to the main distribution.
- Perl has been tested with the GNU pth userlevel thread package (http://www.gnu.org/software/pth/pth.html). All thread tests of Perl now work, but not without adding some yield()s to the tests, so while pth (and other userlevel thread implementations) can be considered to be "working" with Perl ithreads, keep in mind the possible non-preemptability of the underlying thread implementation.
- Stratus VOS is now supported using Perl's native build method (Configure). This is the recommended method to build Perl on VOS. The older methods, which build miniperl, are still available. See *perlvos*. [561+]
- The Amdahl UTS Unix mainframe platform is now supported. [561]
- WinCE is now supported. See *perlce*.
- z/OS (formerly known as OS/390, formerly known as MVS OE) now has support for dynamic loading. This is not selected by default, however, you must specify -Dusedl in the arguments of Configure. [561]

Selected Bug Fixes

Numerous memory leaks and uninitialized memory accesses have been hunted down. Most importantly, anonymous subs used to leak quite a bit. [561]

- The autouse pragma didn't work for Multi::Part::Function::Names.
- caller() could cause core dumps in certain situations. Carp was sometimes affected by this problem. In particular, caller() now returns a subroutine name of `(unknown)` for subroutines that have been removed from the symbol table.
- chop(@list) in list context returned the characters chopped in reverse order. This has been

reversed to be in the right order. [561]

- Configure no longer includes the DBM libraries (dbm, gdbm, db, ndbm) when building the Perl binary. The only exception to this is SunOS 4.x, which needs them. [561]
- The behaviour of non-decimal but numeric string constants such as "0x23" was platform-dependent: in some platforms that was seen as 35, in some as 0, in some as a floating point number (don't ask). This was caused by Perl's using the operating system libraries in a situation where the result of the string to number conversion is undefined: now Perl consistently handles such strings as zero in numeric contexts.
- Several debugger fixes: exit code now reflects the script exit code, condition `"0"` now treated correctly, the `d` command now checks line number, `$.` no longer gets corrupted, and all debugger output now goes correctly to the socket if RemotePort is set. [561]
- The debugger (perl5db.pl) has been modified to present a more consistent commands interface, via (CommandSet=580). perl5db.t was also added to test the changes, and as a placeholder for further tests.

  See *perldebug*.
- The debugger has a new `dumpDepth` option to control the maximum depth to which nested structures are dumped. The `x` command has been extended so that `x N EXPR` dumps out the value of *EXPR* to a depth of at most *N* levels.
- The debugger can now show lexical variables if you have the CPAN module PadWalker installed.
- The order of DESTROYs has been made more predictable.
- Perl 5.6.0 could emit spurious warnings about redefinition of dl_error() when statically building extensions into perl. This has been corrected. [561]
- *dprofpp* -R didn't work.
- `*foo{FORMAT}` now works.
- Infinity is now recognized as a number.
- UNIVERSAL::isa no longer caches methods incorrectly. (This broke the Tk extension with 5.6.0.) [561]
- Lexicals I: lexicals outside an eval "" weren't resolved correctly inside a subroutine definition inside the eval "" if they were not already referenced in the top level of the eval""ed code.
- Lexicals II: lexicals leaked at file scope into subroutines that were declared before the lexicals.
- Lexical warnings now propagating correctly between scopes and into `eval "..."`.
- `use warnings qw(FATAL all)` did not work as intended. This has been corrected. [561]
- warnings::enabled() now reports the state of $^W correctly if the caller isn't using lexical warnings. [561]
- Line renumbering with eval and `#line` now works. [561]
- Fixed numerous memory leaks, especially in eval "".
- Localised tied variables no longer leak memory

```
use Tie::Hash;
tie my %tied_hash => 'Tie::StdHash';

...
```

```
# Used to leak memory every time local() was called;
# in a loop, this added up.
local($tied_hash{Foo}) = 1;
```

- Localised hash elements (and %ENV) are correctly unlocalised to not exist, if they didn't before they were localised.

```
use Tie::Hash;
tie my %tied_hash => 'Tie::StdHash';

...

# Nothing has set the FOO element so far

{ local $tied_hash{FOO} = 'Bar' }

# This used to print, but not now.
print "exists!\n" if exists $tied_hash{FOO};
```

  As a side effect of this fix, tied hash interfaces **must** define the EXISTS and DELETE methods.

- mkdir() now ignores trailing slashes in the directory name, as mandated by POSIX.
- Some versions of glibc have a broken modfl(). This affects builds with `-Duselongdouble`. This version of Perl detects this brokenness and has a workaround for it. The glibc release 2.2.2 is known to have fixed the modfl() bug.
- Modulus of unsigned numbers now works (4063328477 % 65535 used to return 27406, instead of 27047). [561]
- Some "not a number" warnings introduced in 5.6.0 eliminated to be more compatible with 5.005. Infinity is now recognised as a number. [561]
- Numeric conversions did not recognize changes in the string value properly in certain circumstances. [561]
- Attributes (such as :shared) didn't work with our().
- our() variables will not cause bogus "Variable will not stay shared" warnings. [561]
- "our" variables of the same name declared in two sibling blocks resulted in bogus warnings about "redeclaration" of the variables. The problem has been corrected. [561]
- pack "Z" now correctly terminates the string with "\0".
- Fix password routines which in some shadow password platforms (e.g. HP-UX) caused getpwent() to return every other entry.
- The PERL5OPT environment variable (for passing command line arguments to Perl) didn't work for more than a single group of options. [561]
- PERL5OPT with embedded spaces didn't work.
- printf() no longer resets the numeric locale to "C".
- `qw(a\\b)` now parses correctly as `'a\\b'`: that is, as three characters, not four. [561]
- pos() did not return the correct value within s///ge in earlier versions. This is now handled correctly. [561]
- Printing quads (64-bit integers) with printf/sprintf now works without the q L ll prefixes

(assuming you are on a quad-capable platform).

- Regular expressions on references and overloaded scalars now work. [561+]
- Right-hand side magic (GMAGIC) could in many cases such as string concatenation be invoked too many times.
- scalar() now forces scalar context even when used in void context.
- SOCKS support is now much more robust.
- sort() arguments are now compiled in the right wantarray context (they were accidentally using the context of the sort() itself). The comparison block is now run in scalar context, and the arguments to be sorted are always provided list context. [561]
- Changed the POSIX character class `[[:space:]]` to include the (very rarely used) vertical tab character. Added a new POSIX-ish character class `[[:blank:]]` which stands for horizontal whitespace (currently, the space and the tab).
- The tainting behaviour of sprintf() has been rationalized. It does not taint the result of floating point formats anymore, making the behaviour consistent with that of string interpolation. [561]
- Some cases of inconsistent taint propagation (such as within hash values) have been fixed.
- The RE engine found in Perl 5.6.0 accidentally pessimised certain kinds of simple pattern matches. These are now handled better. [561]
- Regular expression debug output (whether through `use re 'debug'` or via `-Dr`) now looks better. [561]
- Multi-line matches like `"a\nxb\n" =~ /(?!\A)x/m` were flawed. The bug has been fixed. [561]
- Use of $& could trigger a core dump under some situations. This is now avoided. [561]
- The regular expression captured submatches ($1, $2, ...) are now more consistently unset if the match fails, instead of leaving false data lying around in them. [561]
- readline() on files opened in "slurp" mode could return an extra "" (blank line) at the end in certain situations. This has been corrected. [561]
- Autovivification of symbolic references of special variables described in *perlvar* (as in `${$num}`) was accidentally disabled. This works again now. [561]
- Sys::Syslog ignored the `LOG_AUTH` constant.
- $AUTOLOAD, sort(), lock(), and spawning subprocesses in multiple threads simultaneously are now thread-safe.
- Tie::Array's SPLICE method was broken.
- Allow a read-only string on the left-hand side of a non-modifying tr///.
- If `STDERR` is tied, warnings caused by `warn` and `die` now correctly pass to it.
- Several Unicode fixes.
    - BOMs (byte order marks) at the beginning of Perl files (scripts, modules) should now be transparently skipped. UTF-16 and UCS-2 encoded Perl files should now be read correctly.
    - The character tables have been updated to Unicode 3.2.0.
    - Comparing with utf8 data does not magically upgrade non-utf8 data into utf8.

(This was a problem for example if you were mixing data from I/O and Unicode data: your output might have got magically encoded as UTF-8.)

- Generating illegal Unicode code points such as U+FFFE, or the UTF-16 surrogates, now also generates an optional warning.
- `IsAlnum`, `IsAlpha`, and `IsWord` now match titlecase.
- Concatenation with the `.` operator or via variable interpolation, `eq`, `substr`, `reverse`, `quotemeta`, the `x` operator, substitution with `s///`, single-quoted UTF-8, should now work.
- The `tr///` operator now works. Note that the `tr///CU` functionality has been removed (but see pack('U0', ...)).
- `eval "v200"` now works.
- Perl 5.6.0 parsed m/\x{ab}/ incorrectly, leading to spurious warnings. This has been corrected. [561]
- Zero entries were missing from the Unicode classes such as `IsDigit`.

- Large unsigned numbers (those above 2**31) could sometimes lose their unsignedness, causing bogus results in arithmetic operations. [561]
- The Perl parser has been stress tested using both random input and Markov chain input and the few found crashes and lockups have been fixed.

Platform Specific Changes and Fixes

- BSDI 4.*

  Perl now works on post-4.0 BSD/OSes.

- All BSDs

  Setting `$0` now works (as much as possible; see *perlvar* for details).

- Cygwin

  Numerous updates; currently synchronised with Cygwin 1.3.10.

- Previously DYNIX/ptx had problems in its Configure probe for non-blocking I/O.

- EPOC

  EPOC now better supported. See README.epoc. [561]

- FreeBSD 3.*

  Perl now works on post-3.0 FreeBSDs.

- HP-UX

  README.hpux updated; `Configure -Duse64bitall` now works; now uses HP-UX malloc instead of Perl malloc.

- IRIX

  Numerous compilation flag and hint enhancements; accidental mixing of 32-bit and 64-bit libraries (a doomed attempt) made much harder.

- Linux
  - Long doubles should now work (see INSTALL). [561]
  - Linux previously had problems related to sockaddrlen when using accept(), recvfrom() (in Perl: recv()), getpeername(), and getsockname().

- Mac OS Classic

  Compilation of the standard Perl distribution in Mac OS Classic should now work if you have the Metrowerks development environment and the missing Mac-specific toolkit bits. Contact the macperl mailing list for details.

- MPE/iX

  MPE/iX update after Perl 5.6.0. See README.mpeix. [561]

- NetBSD/threads: try installing the GNU pth (should be in the packages collection, or http://www.gnu.org/software/pth/), and Configure with -Duseithreads.

- NetBSD/sparc

  Perl now works on NetBSD/sparc.

- OS/2

  Now works with usethreads (see INSTALL). [561]

- Solaris

  64-bitness using the Sun Workshop compiler now works.

- Stratus VOS

  The native build method requires at least VOS Release 14.5.0 and GNU C++/GNU Tools 2.0.1 or later. The Perl pack function now maps overflowed values to +infinity and underflowed values to -infinity.

- Tru64 (aka Digital UNIX, aka DEC OSF/1)

  The operating system version letter now recorded in $Config{osvers}. Allow compiling with gcc (previously explicitly forbidden). Compiling with gcc still not recommended because buggy code results, even with gcc 2.95.2.

- Unicos

  Fixed various alignment problems that lead into core dumps either during build or later; no longer dies on math errors at runtime; now using full quad integers (64 bits), previously was using only 46 bit integers for speed.

- VMS

  See *Socket Extension Dynamic in VMS* and *IEEE-format Floating Point Default on OpenVMS Alpha* for important changes not otherwise listed here.

  chdir() now works better despite a CRT bug; now works with MULTIPLICITY (see INSTALL); now works with Perl's malloc.

  The tainting of `%ENV` elements via `keys` or `values` was previously unimplemented. It now works as documented.

  The `waitpid` emulation has been improved. The worst bug (now fixed) was that a pid of -1 would cause a wildcard search of all processes on the system.

  POSIX-style signals are now emulated much better on VMS versions prior to 7.0.

  The `system` function and backticks operator have improved functionality and better error handling. [561]

  File access tests now use current process privileges rather than the user's default privileges, which could sometimes result in a mismatch between reported access and actual access. This improvement is only available on VMS v6.0 and later.

  There is a new `kill` implementation based on `sys$sigprc` that allows older VMS systems (pre-7.0) to use `kill` to send signals rather than simply force exit. This implementation also allows later systems to call `kill` from within a signal handler.

Iterative logical name translations are now limited to 10 iterations in imitation of SHOW LOGICAL and other OpenVMS facilities.

- Windows
    - Signal handling now works better than it used to. It is now implemented using a Windows message loop, and is therefore less prone to random crashes.
    - fork() emulation is now more robust, but still continues to have a few esoteric bugs and caveats. See *perlfork* for details. [561+]
    - A failed (pseudo)fork now returns undef and sets errno to EAGAIN. [561]
    - The following modules now work on Windows:

        ```
        ExtUtils::Embed         [561]
        IO::Pipe
        IO::Poll
        Net::Ping
        ```

    - IO::File::new_tmpfile() is no longer limited to 32767 invocations per-process.
    - Better chdir() return value for a non-existent directory.
    - Compiling perl using the 64-bit Platform SDK tools is now supported.
    - The Win32::SetChildShowWindow() builtin can be used to control the visibility of windows created by child processes. See *Win32* for details.
    - Non-blocking waits for child processes (or pseudo-processes) are supported via `waitpid($pid, &POSIX::WNOHANG)`.
    - The behavior of system() with multiple arguments has been rationalized. Each unquoted argument will be automatically quoted to protect whitespace, and any existing whitespace in the arguments will be preserved. This improves the portability of system(@args) by avoiding the need for Windows `cmd` shell specific quoting in perl programs.

      Note that this means that some scripts that may have relied on earlier buggy behavior may no longer work correctly. For example, `system("nmake /nologo", @args)` will now attempt to run the file `nmake /nologo` and will fail when such a file isn't found. On the other hand, perl will now execute code such as `system("c:/Program Files/MyApp/foo.exe", @args)` correctly.
    - The perl header files no longer suppress common warnings from the Microsoft Visual C++ compiler. This means that additional warnings may now show up when compiling XS code.
    - Borland C++ v5.5 is now a supported compiler that can build Perl. However, the generated binaries continue to be incompatible with those generated by the other supported compilers (GCC and Visual C++). [561]
    - Duping socket handles with open(F, ">&MYSOCK") now works under Windows 9x. [561]
    - Current directory entries in %ENV are now correctly propagated to child processes. [561]
    - New %ENV entries now propagate to subprocesses. [561]
    - Win32::GetCwd() correctly returns C:\ instead of C: when at the drive root. Other bugs in chdir() and Cwd::cwd() have also been fixed. [561]

- The makefiles now default to the features enabled in ActiveState ActivePerl (a popular Win32 binary distribution). [561]
- HTML files will now be installed in c:\perl\html instead of c:\perl\lib\pod\html
- REG_EXPAND_SZ keys are now allowed in registry settings used by perl. [561]
- Can now send() from all threads, not just the first one. [561]
- ExtUtils::MakeMaker now uses $ENV{LIB} to search for libraries. [561]
- Less stack reserved per thread so that more threads can run concurrently. (Still 16M per thread.) [561]
- `File::Spec->tmpdir()` now prefers C:/temp over /tmp (works better when perl is running as service).
- Better UNC path handling under ithreads. [561]
- wait(), waitpid(), and backticks now return the correct exit status under Windows 9x. [561]
- A socket handle leak in accept() has been fixed. [561]

New or Changed Diagnostics

Please see *perldiag* for more details.

- Ambiguous range in the transliteration operator (like a-z-9) now gives a warning.
- chdir("") and chdir(undef) now give a deprecation warning because they cause a possible unintentional chdir to the home directory. Say chdir() if you really mean that.
- Two new debugging options have been added: if you have compiled your Perl with debugging, you can use the -DT [561] and -DR options to trace tokenising and to add reference counts to displaying variables, respectively.
- The lexical warnings category "deprecated" is no longer a sub-category of the "syntax" category. It is now a top-level category in its own right.
- Unadorned dump() will now give a warning suggesting to use explicit CORE::dump() if that's what really is meant.
- The "Unrecognized escape" warning has been extended to include `\8`, `\9`, and `\_`. There is no need to escape any of the `\w` characters.
- All regular expression compilation error messages are now hopefully easier to understand both because the error message now comes before the failed regex and because the point of failure is now clearly marked by a `<-- HERE` marker.
- Various I/O (and socket) functions like binmode(), close(), and so forth now more consistently warn if they are used illogically either on a yet unopened or on an already closed filehandle (or socket).
- Using lstat() on a filehandle now gives a warning. (It's a non-sensical thing to do.)
- The `-M` and `-m` options now warn if you didn't supply the module name.
- If you in `use` specify a required minimum version, modules matching the name and but not defining a $VERSION will cause a fatal failure.
- Using negative offset for vec() in lvalue context is now a warnable offense.

- Odd number of arguments to overload::constant now elicits a warning.
- Odd number of elements in anonymous hash now elicits a warning.
- The various "opened only for", "on closed", "never opened" warnings drop the `main::` prefix for filehandles in the `main` package, for example `STDIN` instead of `main::STDIN`.
- Subroutine prototypes are now checked more carefully, you may get warnings for example if you have used non-prototype characters.
- If an attempt to use a (non-blessed) reference as an array index is made, a warning is given.
- `push @a;` and `unshift @a;` (with no values to push or unshift) now give a warning. This may be a problem for generated and eval'ed code.
- If you try to *"pack" in perlfunc* a number less than 0 or larger than 255 using the `"C"` format you will get an optional warning. Similarly for the `"c"` format and a number less than -128 or more than 127.
- pack `P` format now demands an explicit size.
- unpack `w` now warns of unterminated compressed integers.
- Warnings relating to the use of PerlIO have been added.
- Certain regex modifiers such as `(?o)` make sense only if applied to the entire regex. You will get an optional warning if you try to do otherwise.
- Variable length lookbehind has not yet been implemented, trying to use it will tell that.
- Using arrays or hashes as references (e.g. `%foo->{bar}` has been deprecated for a while. Now you will get an optional warning.
- Warnings relating to the use of the new restricted hashes feature have been added.
- Self-ties of arrays and hashes are not supported and fatal errors will happen even at an attempt to do so.
- Using `sort` in scalar context now issues an optional warning. This didn't do anything useful, as the sort was not performed.
- Using the /g modifier in split() is meaningless and will cause a warning.
- Using splice() past the end of an array now causes a warning.
- Malformed Unicode encodings (UTF-8 and UTF-16) cause a lot of warnings, as does trying to use UTF-16 surrogates (which are unimplemented).
- Trying to use Unicode characters on an I/O stream without marking the stream's encoding (using open() or binmode()) will cause "Wide character" warnings.
- Use of v-strings in use/require causes a (backward) portability warning.
- Warnings relating to the use interpreter threads and their shared data have been added.

Changed Internals

- PerlIO is now the default.
- perlapi.pod (a companion to perlguts) now attempts to document the internal API.
- You can now build a really minimal perl called microperl. Building microperl does not require even running Configure; `make -f Makefile.micro` should be enough. Beware: microperl makes many assumptions, some of which may be too bold; the resulting executable may crash or otherwise misbehave in wondrous ways. For careful hackers only.

- Added rsignal(), whichsig(), do_join(), op_clear, op_null, ptr_table_clear(), ptr_table_free(), sv_setref_uv(), and several UTF-8 interfaces to the publicised API. For the full list of the available APIs see *perlapi*.
- Made possible to propagate customised exceptions via croak()ing.
- Now xsubs can have attributes just like subs. (Well, at least the built-in attributes.)
- dTHR and djSP have been obsoleted; the former removed (because it's a no-op) and the latter replaced with dSP.
- PERL_OBJECT has been completely removed.
- The MAGIC constants (e.g. `'P'`) have been macrofied (e.g. `PERL_MAGIC_TIED`) for better source code readability and maintainability.
- The regex compiler now maintains a structure that identifies nodes in the compiled bytecode with the corresponding syntactic features of the original regex expression. The information is attached to the new `offsets` member of the `struct regexp`. See *perldebguts* for more complete information.
- The C code has been made much more `gcc -Wall` clean. Some warning messages still remain in some platforms, so if you are compiling with gcc you may see some warnings about dubious practices. The warnings are being worked on.
- *perly.c*, *sv.c*, and *sv.h* have now been extensively commented.
- Documentation on how to use the Perl source repository has been added to *Porting/repository.pod*.
- There are now several profiling make targets.

Security Vulnerability Closed [561]

(This change was already made in 5.7.0 but bears repeating here.) (5.7.0 came out before 5.6.1: the development branch 5.7 released earlier than the maintenance branch 5.6)

A potential security vulnerability in the optional suidperl component of Perl was identified in August 2000. suidperl is neither built nor installed by default. As of November 2001 the only known vulnerable platform is Linux, most likely all Linux distributions. CERT and various vendors and distributors have been alerted about the vulnerability. See http://www.cpan.org/src/5.0/sperl-2000-08-05/sperl-2000-08-05.txt for more information.

The problem was caused by Perl trying to report a suspected security exploit attempt using an external program, /bin/mail. On Linux platforms the /bin/mail program had an undocumented feature which when combined with suidperl gave access to a root shell, resulting in a serious compromise instead of reporting the exploit attempt. If you don't have /bin/mail, or if you have 'safe setuid scripts', or if suidperl is not installed, you are safe.

The exploit attempt reporting feature has been completely removed from Perl 5.8.0 (and the maintenance release 5.6.1, and it was removed also from all the Perl 5.7 releases), so that particular vulnerability isn't there anymore. However, further security vulnerabilities are, unfortunately, always possible. The suidperl functionality is most probably going to be removed in Perl 5.10. In any case, suidperl should only be used by security experts who know exactly what they are doing and why they are using suidperl instead of some other solution such as sudo (see http://www.courtesan.com/sudo/).

New Tests

Several new tests have been added, especially for the *lib* and *ext* subsections. There are now about 69 000 individual tests (spread over about 700 test scripts), in the regression suite (5.6.1 has about 11 700 tests, in 258 test scripts) The exact numbers depend on the platform and Perl configuration

used. Many of the new tests are of course introduced by the new modules, but still in general Perl is now more thoroughly tested.

Because of the large number of tests, running the regression suite will take considerably longer time than it used to: expect the suite to take up to 4-5 times longer to run than in perl 5.6. On a really fast machine you can hope to finish the suite in about 6-8 minutes (wallclock time).

The tests are now reported in a different order than in earlier Perls. (This happens because the test scripts from under t/lib have been moved to be closer to the library/extension they are testing.)

Known Problems

The Compiler Suite Is Still Very Experimental

The compiler suite is slowly getting better but it continues to be highly experimental. Use in production environments is discouraged.

Localising Tied Arrays and Hashes Is Broken

```
local %tied_array;
```

doesn't work as one would expect: the old value is restored incorrectly. This will be changed in a future release, but we don't know yet what the new semantics will exactly be. In any case, the change will break existing code that relies on the current (ill-defined) semantics, so just avoid doing this in general.

Building Extensions Can Fail Because Of Largefiles

Some extensions like mod_perl are known to have issues with `largefiles', a change brought by Perl 5.6.0 in which file offsets default to 64 bits wide, where supported. Modules may fail to compile at all, or they may compile and work incorrectly. Currently, there is no good solution for the problem, but Configure now provides appropriate non-largefile ccflags, ldflags, libswanted, and libs in the %Config hash (e.g., $Config{ccflags_nolargefiles}) so the extensions that are having problems can try configuring themselves without the largefileness. This is admittedly not a clean solution, and the solution may not even work at all. One potential failure is whether one can (or, if one can, whether it's a good idea to) link together at all binaries with different ideas about file offsets; all this is platform-dependent.

Modifying $_ Inside for(..)

```
for (1..5) { $_++ }
```

works without complaint. It shouldn't. (You should be able to modify only lvalue elements inside the loops.) You can see the correct behaviour by replacing the 1..5 with 1, 2, 3, 4, 5.

mod_perl 1.26 Doesn't Build With Threaded Perl

Use mod_perl 1.27 or higher.

lib/ftmp-security tests warn 'system possibly insecure'

Don't panic. Read the 'make test' section of INSTALL instead.

libwww-perl (LWP) fails base/date #51

Use libwww-perl 5.65 or later.

PDL failing some tests

Use PDL 2.3.4 or later.

Perl_get_sv

You may get errors like 'Undefined symbol "Perl_get_sv"' or "can't resolve symbol 'Perl_get_sv'", or the symbol may be "Perl_sv_2pv". This probably means that you are trying to use an older shared Perl library (or extensions linked with such) with Perl 5.8.0 executable. Perl used to have such a

subroutine, but that is no more the case. Check your shared library path, and any shared Perl libraries in those directories.

Sometimes this problem may also indicate a partial Perl 5.8.0 installation, see *Mac OS X dyld undefined symbols* for an example and how to deal with it.

Self-tying Problems

Self-tying of arrays and hashes is broken in rather deep and hard-to-fix ways. As a stop-gap measure to avoid people from getting frustrated at the mysterious results (core dumps, most often), it is forbidden for now (you will get a fatal error even from an attempt).

A change to self-tying of globs has caused them to be recursively referenced (see: *"Two-Phased Garbage Collection" in perlobj*). You will now need an explicit untie to destroy a self-tied glob. This behaviour may be fixed at a later date.

Self-tying of scalars and IO thingies works.

ext/threads/t/libc

If this test fails, it indicates that your libc (C library) is not threadsafe. This particular test stress tests the localtime() call to find out whether it is threadsafe. See *perlthrtut* for more information.

Failure of Thread (5.005-style) tests

Note that support for 5.005-style threading is deprecated, experimental and practically unsupported. In 5.10, it is expected to be removed. You should migrate your code to ithreads.

The following tests are known to fail due to fundamental problems in the 5.005 threading implementation. These are not new failures--Perl 5.005_0x has the same bugs, but didn't have these tests.

```
 ../ext/B/t/xref.t                    255 65280    14   12  85.71%  3-14
 ../ext/List/Util/t/first.t           255 65280     7    4  57.14%  2 5-7
 ../lib/English.t                       2   512    54    2   3.70%  2-3
 ../lib/FileCache.t                                 5    1  20.00%  5
 ../lib/Filter/Simple/t/data.t                      6    3  50.00%  1-3
 ../lib/Filter/Simple/t/filter_only.                9    3  33.33%  1-2 5
 ../lib/Math/BigInt/t/bare_mbf.t                 1627    4   0.25%  8 11
1626-1627
 ../lib/Math/BigInt/t/bigfltpm.t                 1629    4   0.25%  10 13
1628-
                                                                    1629
 ../lib/Math/BigInt/t/sub_mbf.t                  1633    4   0.24%  8 11
1632-1633
 ../lib/Math/BigInt/t/with_sub.t                 1628    4   0.25%  9 12
1627-1628
 ../lib/Tie/File/t/31_autodefer.t     255 65280    65   32  49.23%  34-65
 ../lib/autouse.t                                  10    1  10.00%  4
 op/flip.t                                         15    1   6.67%  15
```

These failures are unlikely to get fixed as 5.005-style threads are considered fundamentally broken. (Basically what happens is that competing threads can corrupt shared global state, one good example being regular expression engine's state.)

Timing problems

The following tests may fail intermittently because of timing problems, for example if the system is heavily loaded.

```
    t/op/alarm.t
    ext/Time/HiRes/HiRes.t
```

```
lib/Benchmark.t
lib/Memoize/t/expmod_t.t
lib/Memoize/t/speed.t
```

In case of failure please try running them manually, for example

```
./perl -Ilib ext/Time/HiRes/HiRes.t
```

Tied/Magical Array/Hash Elements Do Not Autovivify

For normal arrays `$foo = \$bar[1]` will assign `undef` to `$bar[1]` (assuming that it didn't exist before), but for tied/magical arrays and hashes such autovivification does not happen because there is currently no way to catch the reference creation. The same problem affects slicing over non-existent indices/keys of a tied/magical array/hash.

Unicode in package/class and subroutine names does not work

One can have Unicode in identifier names, but not in package/class or subroutine names. While some limited functionality towards this does exist as of Perl 5.8.0, that is more accidental than designed; use of Unicode for the said purposes is unsupported.

One reason of this unfinishedness is its (currently) inherent unportability: since both package names and subroutine names may need to be mapped to file and directory names, the Unicode capability of the filesystem becomes important-- and there unfortunately aren't portable answers.

Platform Specific Problems

AIX

- If using the AIX native make command, instead of just "make" issue "make all". In some setups the former has been known to spuriously also try to run "make install". Alternatively, you may want to use GNU make.
- In AIX 4.2, Perl extensions that use C++ functions that use statics may have problems in that the statics are not getting initialized. In newer AIX releases, this has been solved by linking Perl with the libC_r library, but unfortunately in AIX 4.2 the said library has an obscure bug where the various functions related to time (such as time() and gettimeofday()) return broken values, and therefore in AIX 4.2 Perl is not linked against libC_r.
- vac 5.0.0.0 May Produce Buggy Code For Perl

  The AIX C compiler vac version 5.0.0.0 may produce buggy code, resulting in a few random tests failing when run as part of "make test", but when the failing tests are run by hand, they succeed. We suggest upgrading to at least vac version 5.0.1.0, that has been known to compile Perl correctly. "lslpp -L|grep vac.C" will tell you the vac version. See README.aix.
- If building threaded Perl, you may get compilation warning from pp_sys.c:

  ```
  "pp_sys.c", line 4651.39: 1506-280 (W) Function argument assignment
  between types "unsigned char*" and "const void*" is not allowed.
  ```

  This is harmless; it is caused by the getnetbyaddr() and getnetbyaddr_r() having slightly different types for their first argument.

Alpha systems with old gccs fail several tests

If you see op/pack, op/pat, op/regexp, or ext/Storable tests failing in a Linux/alpha or *BSD/Alpha, it's probably time to upgrade your gcc. gccs prior to 2.95.3 are definitely not good enough, and gcc 3.1 may be even better. (RedHat Linux/alpha with gcc 3.1 reported no problems, as did Linux 2.4.18 with gcc 2.95.4.) (In Tru64, it is preferable to use the bundled C compiler.)

AmigaOS

Perl 5.8.0 doesn't build in AmigaOS. It broke at some point during the ithreads work and we could not find Amiga experts to unbreak the problems. Perl 5.6.1 still works for AmigaOS (as does the 5.7.2 development release).

BeOS

The following tests fail on 5.8.0 Perl in BeOS Personal 5.03:

```
t/op/lfs.............................FAILED at test 17
t/op/magic...........................FAILED at test 24
ext/Fcntl/t/syslfs...................FAILED at test 17
ext/File/Glob/t/basic................FAILED at test 3
ext/POSIX/t/sigaction................FAILED at test 13
ext/POSIX/t/waitpid..................FAILED at test 1
```

(**Note:** more information was available in *README.beos* until support for BeOS was removed in Perl v5.18.0)

Cygwin "unable to remap"

For example when building the Tk extension for Cygwin, you may get an error message saying "unable to remap". This is known problem with Cygwin, and a workaround is detailed in here: http://sources.redhat.com/ml/cygwin/2001-12/msg00894.html

Cygwin ndbm tests fail on FAT

One can build but not install (or test the build of) the NDBM_File on FAT filesystems. Installation (or build) on NTFS works fine. If one attempts the test on a FAT install (or build) the following failures are expected:

```
../ext/NDBM_File/ndbm.t       13  3328    71   59  83.10%  1-2 4 16-71
../ext/ODBM_File/odbm.t      255 65280    ??   ??       %  ??
../lib/AnyDBM_File.t           2   512    12    2  16.67%  1 4
../lib/Memoize/t/errors.t      0   139    11    5  45.45%  7-11
../lib/Memoize/t/tie_ndbm.t   13  3328     4    4 100.00%  1-4
run/fresh_perl.t                       97    1   1.03%  91
```

NDBM_File fails and ODBM_File just coredumps.

If you intend to run only on FAT (or if using AnyDBM_File on FAT), run Configure with the -Ui_ndbm and -Ui_dbm options to prevent NDBM_File and ODBM_File being built.

DJGPP Failures

```
t/op/stat............................FAILED at test 29
lib/File/Find/t/find.................FAILED at test 1
lib/File/Find/t/taint................FAILED at test 1
lib/h2xs.............................FAILED at test 15
lib/Pod/t/eol........................FAILED at test 1
lib/Test/Harness/t/strap-analyze.....FAILED at test 8
lib/Test/Harness/t/test-harness......FAILED at test 23
lib/Test/Simple/t/exit...............FAILED at test 1
```

The above failures are known as of 5.8.0 with native builds with long filenames, but there are a few more if running under dosemu because of limitations (and maybe bugs) of dosemu:

```
t/comp/cpp...........................FAILED at test 3
t/op/inccode.........................(crash)
```

and a few lib/ExtUtils tests, and several hundred Encode/t/Aliases.t failures that work fine with long filenames. So you really might prefer native builds and long filenames.

FreeBSD built with ithreads coredumps reading large directories

This is a known bug in FreeBSD 4.5's readdir_r(), it has been fixed in FreeBSD 4.6 (see *perlfreebsd* (README.freebsd)).

FreeBSD Failing locale Test 117 For ISO 8859-15 Locales

The ISO 8859-15 locales may fail the locale test 117 in FreeBSD. This is caused by the characters \xFF (y with diaeresis) and \xBE (Y with diaeresis) not behaving correctly when being matched case-insensitively. Apparently this problem has been fixed in the latest FreeBSD releases. (http://www.freebsd.org/cgi/query-pr.cgi?pr=34308)

IRIX fails ext/List/Util/t/shuffle.t or Digest::MD5

IRIX with MIPSpro 7.3.1.2m or 7.3.1.3m compiler may fail the List::Util test ext/List/Util/t/shuffle.t by dumping core. This seems to be a compiler error since if compiled with gcc no core dump ensues, and no failures have been seen on the said test on any other platform.

Similarly, building the Digest::MD5 extension has been known to fail with "*** Termination code 139 (bu21)".

The cure is to drop optimization level (Configure -Doptimize=-O2).

HP-UX lib/posix Subtest 9 Fails When LP64-Configured

If perl is configured with -Duse64bitall, the successful result of the subtest 10 of lib/posix may arrive before the successful result of the subtest 9, which confuses the test harness so much that it thinks the subtest 9 failed.

Linux with glibc 2.2.5 fails t/op/int subtest #6 with -Duse64bitint

This is a known bug in the glibc 2.2.5 with long long integers. (http://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=65612)

Linux With Sfio Fails op/misc Test 48

No known fix.

Mac OS X

Please remember to set your environment variable LC_ALL to "C" (setenv LC_ALL C) before running "make test" to avoid a lot of warnings about the broken locales of Mac OS X.

The following tests are known to fail in Mac OS X 10.1.5 because of buggy (old) implementations of Berkeley DB included in Mac OS X:

```
 Failed Test                Stat Wstat Total Fail  Failed  List of Failed
 -------------------------------------------------------------------------
 ../ext/DB_File/t/db-btree.t    0    11    ??   ??       %  ??
 ../ext/DB_File/t/db-recno.t              149    3   2.01%  61 63 65
```

If you are building on a UFS partition, you will also probably see t/op/stat.t subtest #9 fail. This is caused by Darwin's UFS not supporting inode change time.

Also the ext/POSIX/t/posix.t subtest #10 fails but it is skipped for now because the failure is Apple's fault, not Perl's (blocked signals are lost).

If you Configure with ithreads, ext/threads/t/libc.t will fail. Again, this is not Perl's fault-- the libc of Mac OS X is not threadsafe (in this particular test, the localtime() call is found to be threadunsafe.)

Mac OS X dyld undefined symbols

If after installing Perl 5.8.0 you are getting warnings about missing symbols, for example

```
dyld: perl Undefined symbols
_perl_sv_2pv
_perl_get_sv
```

you probably have an old pre-Perl-5.8.0 installation (or parts of one) in /Library/Perl (the undefined symbols used to exist in pre-5.8.0 Perls). It seems that for some reason "make install" doesn't always completely overwrite the files in /Library/Perl. You can move the old Perl shared library out of the way like this:

```
cd /Library/Perl/darwin/CORE
mv libperl.dylib libperlold.dylib
```

and then reissue "make install". Note that the above of course is extremely disruptive for anything using the /usr/local/bin/perl. If that doesn't help, you may have to try removing all the .bundle files from beneath /Library/Perl, and again "make install"-ing.

OS/2 Test Failures

The following tests are known to fail on OS/2 (for clarity only the failures are shown, not the full error messages):

```
../lib/ExtUtils/t/Mkbootstrap.t    1   256    18    1   5.56%  8
../lib/ExtUtils/t/Packlist.t       1   256    34    1   2.94%  17
../lib/ExtUtils/t/basic.t          1   256    17    1   5.88%  14
lib/os2_process.t                  2   512   227    2   0.88%  174 209
lib/os2_process_kid.t                    227    2   0.88%  174 209
lib/rx_cmprt.t                   255 65280    18    3  16.67%  16-18
```

op/sprintf tests 91, 129, and 130

The op/sprintf tests 91, 129, and 130 are known to fail on some platforms. Examples include any platform using sfio, and Compaq/Tandem's NonStop-UX.

Test 91 is known to fail on QNX6 (nto), because `sprintf '%e',0` incorrectly produces `0.000000e+0` instead of `0.000000e+00`.

For tests 129 and 130, the failing platforms do not comply with the ANSI C Standard: lines 19ff on page 134 of ANSI X3.159 1989, to be exact. (They produce something other than "1" and "-1" when formatting 0.6 and -0.6 using the printf format "%.0f"; most often, they produce "0" and "-0".)

SCO

The socketpair tests are known to be unhappy in SCO 3.2v5.0.4:

```
ext/Socket/socketpair.t..............FAILED tests 15-45
```

Solaris 2.5

In case you are still using Solaris 2.5 (aka SunOS 5.5), you may experience failures (the test core dumping) in lib/locale.t. The suggested cure is to upgrade your Solaris.

Solaris x86 Fails Tests With -Duse64bitint

The following tests are known to fail in Solaris x86 with Perl configured to use 64 bit integers:

```
ext/Data/Dumper/t/dumper.............FAILED at test 268
ext/Devel/Peek/Peek.................FAILED at test 7
```

SUPER-UX (NEC SX)

The following tests are known to fail on SUPER-UX:

```
op/64bitint.........................FAILED tests 29-30, 32-33, 35-36
op/arith............................FAILED tests 128-130
op/pack.............................FAILED tests 25-5625
op/pow..............................
op/taint............................# msgsnd failed
../ext/IO/lib/IO/t/io_poll..........FAILED tests 3-4
../ext/IPC/SysV/ipcsysv.............FAILED tests 2, 5-6
../ext/IPC/SysV/t/msg...............FAILED tests 2, 4-6
../ext/Socket/socketpair............FAILED tests 12
../lib/IPC/SysV.....................FAILED tests 2, 5-6
../lib/warnings.....................FAILED tests 115-116, 118-119
```

The op/pack failure ("Cannot compress negative numbers at op/pack.t line 126") is serious but as of yet unsolved. It points at some problems with the signedness handling of the C compiler, as do the 64bitint, arith, and pow failures. Most of the rest point at problems with SysV IPC.

Term::ReadKey not working on Win32

Use Term::ReadKey 2.20 or later.

UNICOS/mk

- During Configure, the test

  ```
  Guessing which symbols your C compiler and preprocessor define...
  ```

  will probably fail with error messages like

  ```
  CC-20 cc: ERROR File = try.c, Line = 3
    The identifier "bad" is undefined.

    bad switch yylook 79bad switch yylook 79bad switch yylook 79bad
 switch yylook 79#ifdef A29K
    ^

  CC-65 cc: ERROR File = try.c, Line = 3
    A semicolon is expected at this point.
  ```

  This is caused by a bug in the awk utility of UNICOS/mk. You can ignore the error, but it does cause a slight problem: you cannot fully benefit from the h2ph utility (see *h2ph*) that can be used to convert C headers to Perl libraries, mainly used to be able to access from Perl the constants defined using C preprocessor, cpp. Because of the above error, parts of the converted headers will be invisible. Luckily, these days the need for h2ph is rare.

- If building Perl with interpreter threads (ithreads), the getgrent(), getgrnam(), and getgrgid() functions cannot return the list of the group members due to a bug in the multithreaded support of UNICOS/mk. What this means is that in list context the functions will return only three values, not four.

UTS

There are a few known test failures. (**Note:** the relevant information was available in *README.uts* until support for UTS was removed in Perl v5.18.0)

VOS (Stratus)

When Perl is built using the native build process on VOS Release 14.5.0 and GNU C++/GNU Tools 2.0.1, all attempted tests either pass or result in TODO (ignored) failures.

VMS

There should be no reported test failures with a default configuration, though there are a number of tests marked TODO that point to areas needing further debugging and/or porting work.

Win32

In multi-CPU boxes, there are some problems with the I/O buffering: some output may appear twice.

XML::Parser not working

Use XML::Parser 2.31 or later.

z/OS (OS/390)

z/OS has rather many test failures but the situation is actually much better than it was in 5.6.0; it's just that so many new modules and tests have been added.

```
 Failed Test                   Stat Wstat Total Fail  Failed  List of
Failed

-------------------------------------------------------------------------------
 ../ext/Data/Dumper/t/dumper.t              357    8   2.24%  311 314 325
327
                                                             331 333 337
339
 ../ext/IO/lib/IO/t/io_unix.t                 5    4  80.00%  2-5
 ../ext/Storable/t/downgrade.t   12  3072   169   12   7.10%  14-15 46-47
78-79
                                                             110-111 150
161
 ../lib/ExtUtils/t/Constant.t   121 30976    48   48 100.00%  1-48
 ../lib/ExtUtils/t/Embed.t                    9    9 100.00%  1-9
 op/pat.t                                   922    7   0.76%  665 776 785
832-
                                                             834 845
 op/sprintf.t                               224    3   1.34%  98 100 136
 op/tr.t                                     97    5   5.15%  63 71-74
 uni/fold.t                                 780    6   0.77%  61 169 196
661
                                                             710-711
```

The failures in dumper.t and downgrade.t are problems in the tests, those in io_unix and sprintf are problems in the USS (UDP sockets and printf formats). The pat, tr, and fold failures are genuine Perl problems caused by EBCDIC (and in the pat and fold cases, combining that with Unicode). The Constant and Embed are probably problems in the tests (since they test Perl's ability to build extensions, and that seems to be working reasonably well.)

Unicode Support on EBCDIC Still Spotty

Though mostly working, Unicode support still has problem spots on EBCDIC platforms. One such known spot are the `\p{}` and `\P{}` regular expression constructs for code points less than 256: the `pP` are testing for Unicode code points, not knowing about EBCDIC.

Seen In Perl 5.7 But Gone Now

`Time::Piece` (previously known as `Time::Object`) was removed because it was felt that it didn't have enough value in it to be a core module. It is still a useful module, though, and is available from the CPAN.

Perl 5.8 unfortunately does not build anymore on AmigaOS; this broke accidentally at some point. Since there are not that many Amiga developers available, we could not get this fixed and tested in

time for 5.8.0. Perl 5.6.1 still works for AmigaOS (as does the 5.7.2 development release).

The `PerlIO::Scalar` and `PerlIO::Via` (capitalised) were renamed as `PerlIO::scalar` and `PerlIO::via` (all lowercase) just before 5.8.0. The main rationale was to have all core PerlIO layers to have all lowercase names. The "plugins" are named as usual, for example `PerlIO::via::QuotedPrint`.

The `threads::shared::queue` and `threads::shared::semaphore` were renamed as `Thread::Queue` and `Thread::Semaphore` just before 5.8.0. The main rationale was to have thread modules to obey normal naming, `Thread::` (the `threads` and `threads::shared` themselves are more pragma-like, they affect compile-time, so they stay lowercase).

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://bugs.perl.org/ . There may also be information at http://www.perl.com/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

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

HISTORY

Written by Jarkko Hietaniemi <*jhi@iki.fi*>.