NAME

perl581delta - what is new for perl v5.8.1

DESCRIPTION

This document describes differences between the 5.8.0 release and the 5.8.1 release.

If you are upgrading from an earlier release such as 5.6.1, first read the *perl58delta*, which describes differences between 5.6.0 and 5.8.0.

In case you are wondering about 5.6.1, it was bug-fix-wise rather identical to the development release 5.7.1. Confused? This timeline hopefully helps a bit: it lists the new major releases, their maintenance releases, and the development releases.

```
New     Maintenance  Development

5.6.0                             2000-Mar-22
                     5.7.0        2000-Sep-02
        5.6.1                     2001-Apr-08
                     5.7.1        2001-Apr-09
                     5.7.2        2001-Jul-13
                     5.7.3        2002-Mar-05
5.8.0                             2002-Jul-18
        5.8.1                     2003-Sep-25
```

Incompatible Changes

Hash Randomisation

Mainly due to security reasons, the "random ordering" of hashes has been made even more random. Previously while the order of hash elements from keys(), values(), and each() was essentially random, it was still repeatable. Now, however, the order varies between different runs of Perl.

Perl has never guaranteed any ordering of the hash keys, and the ordering has already changed several times during the lifetime of Perl 5. Also, the ordering of hash keys has always been, and continues to be, affected by the insertion order.

The added randomness may affect applications.

One possible scenario is when output of an application has included hash data. For example, if you have used the Data::Dumper module to dump data into different files, and then compared the files to see whether the data has changed, now you will have false positives since the order in which hashes are dumped will vary. In general the cure is to sort the keys (or the values); in particular for Data::Dumper to use the `Sortkeys` option. If some particular order is really important, use tied hashes: for example the Tie::IxHash module which by default preserves the order in which the hash elements were added.

More subtle problem is reliance on the order of "global destruction". That is what happens at the end of execution: Perl destroys all data structures, including user data. If your destructors (the DESTROY subroutines) have assumed any particular ordering to the global destruction, there might be problems ahead. For example, in a destructor of one object you cannot assume that objects of any other class are still available, unless you hold a reference to them. If the environment variable PERL_DESTRUCT_LEVEL is set to a non-zero value, or if Perl is exiting a spawned thread, it will also destruct the ordinary references and the symbol tables that are no longer in use. You can't call a class method or an ordinary function on a class that has been collected that way.

The hash randomisation is certain to reveal hidden assumptions about some particular ordering of hash elements, and outright bugs: it revealed a few bugs in the Perl core and core modules.

To disable the hash randomisation in runtime, set the environment variable PERL_HASH_SEED to 0

(zero) before running Perl (for more information see *"PERL_HASH_SEED" in perlrun*), or to disable the feature completely in compile time, compile with `-DNO_HASH_SEED` (see *INSTALL*).

See *"Algorithmic Complexity Attacks" in perlsec* for the original rationale behind this change.

UTF-8 On Filehandles No Longer Activated By Locale

In Perl 5.8.0 all filehandles, including the standard filehandles, were implicitly set to be in Unicode UTF-8 if the locale settings indicated the use of UTF-8. This feature caused too many problems, so the feature was turned off and redesigned: see *Core Enhancements*.

Single-number v-strings are no longer v-strings before "=>"

The version strings or v-strings (see *"Version Strings" in perldata*) feature introduced in Perl 5.6.0 has been a source of some confusion-- especially when the user did not want to use it, but Perl thought it knew better. Especially troublesome has been the feature that before a "=>" a version string (a "v" followed by digits) has been interpreted as a v-string instead of a string literal. In other words:

```
%h = ( v65 => 42 );
```

has meant since Perl 5.6.0

```
%h = ( 'A' => 42 );
```

(at least in platforms of ASCII progeny) Perl 5.8.1 restores the more natural interpretation

```
%h = ( 'v65' => 42 );
```

The multi-number v-strings like v65.66 and 65.66.67 still continue to be v-strings in Perl 5.8.

(Win32) The -C Switch Has Been Repurposed

The -C switch has changed in an incompatible way. The old semantics of this switch only made sense in Win32 and only in the "use utf8" universe in 5.6.x releases, and do not make sense for the Unicode implementation in 5.8.0. Since this switch could not have been used by anyone, it has been repurposed. The behavior that this switch enabled in 5.6.x releases may be supported in a transparent, data-dependent fashion in a future release.

For the new life of this switch, see *UTF-8 no longer default under UTF-8 locales*, and *"-C" in perlrun*.

(Win32) The /d Switch Of cmd.exe

Perl 5.8.1 uses the /d switch when running the cmd.exe shell internally for system(), backticks, and when opening pipes to external programs. The extra switch disables the execution of AutoRun commands from the registry, which is generally considered undesirable when running external programs. If you wish to retain compatibility with the older behavior, set PERL5SHELL in your environment to `cmd /x/c`.

Core Enhancements

UTF-8 no longer default under UTF-8 locales

In Perl 5.8.0 many Unicode features were introduced. One of them was found to be of more nuisance than benefit: the automagic (and silent) "UTF-8-ification" of filehandles, including the standard filehandles, if the user's locale settings indicated use of UTF-8.

For example, if you had `en_US.UTF-8` as your locale, your STDIN and STDOUT were automatically "UTF-8", in other words an implicit binmode(..., ":utf8") was made. This meant that trying to print, say, chr(0xff), ended up printing the bytes 0xc3 0xbf. Hardly what you had in mind unless you were aware of this feature of Perl 5.8.0. The problem is that the vast majority of people weren't: for example in RedHat releases 8 and 9 the **default** locale setting is UTF-8, so all RedHat users got UTF-8 filehandles, whether they wanted it or not. The pain was intensified by the Unicode implementation of Perl 5.8.0 (still) having nasty bugs, especially related to the use of s/// and tr///. (Bugs that have been

fixed in 5.8.1)

Therefore a decision was made to backtrack the feature and change it from implicit silent default to explicit conscious option. The new Perl command line option `-C` and its counterpart environment variable PERL_UNICODE can now be used to control how Perl and Unicode interact at interfaces like I/O and for example the command line arguments. See *"-C" in perlrun* and *"PERL_UNICODE" in perlrun* for more information.

Unsafe signals again available

In Perl 5.8.0 the so-called "safe signals" were introduced. This means that Perl no longer handles signals immediately but instead "between opcodes", when it is safe to do so. The earlier immediate handling easily could corrupt the internal state of Perl, resulting in mysterious crashes.

However, the new safer model has its problems too. Because now an opcode, a basic unit of Perl execution, is never interrupted but instead let to run to completion, certain operations that can take a long time now really do take a long time. For example, certain network operations have their own blocking and timeout mechanisms, and being able to interrupt them immediately would be nice.

Therefore perl 5.8.1 introduces a "backdoor" to restore the pre-5.8.0 (pre-5.7.3, really) signal behaviour. Just set the environment variable PERL_SIGNALS to `unsafe`, and the old immediate (and unsafe) signal handling behaviour returns. See *"PERL_SIGNALS" in perlrun* and *"Deferred Signals (Safe Signals)" in perlipc*.

In completely unrelated news, you can now use safe signals with POSIX::SigAction. See *"POSIX::SigAction" in POSIX*.

Tied Arrays with Negative Array Indices

Formerly, the indices passed to `FETCH`, `STORE`, `EXISTS`, and `DELETE` methods in tied array class were always non-negative. If the actual argument was negative, Perl would call FETCHSIZE implicitly and add the result to the index before passing the result to the tied array method. This behaviour is now optional. If the tied array class contains a package variable named `$NEGATIVE_INDICES` which is set to a true value, negative values will be passed to `FETCH`, `STORE`, `EXISTS`, and `DELETE` unchanged.

local ${$x}

The syntaxes

```
local ${$x}
local @{$x}
local %{$x}
```

now do localise variables, given that the $x is a valid variable name.

Unicode Character Database 4.0.0

The copy of the Unicode Character Database included in Perl 5.8 has been updated to 4.0.0 from 3.2.0. This means for example that the Unicode character properties are as in Unicode 4.0.0.

Deprecation Warnings

There is one new feature deprecation. Perl 5.8.0 forgot to add some deprecation warnings, these warnings have now been added. Finally, a reminder of an impending feature removal.

(Reminder) Pseudo-hashes are deprecated (really)

Pseudo-hashes were deprecated in Perl 5.8.0 and will be removed in Perl 5.10.0, see *perl58delta* for details. Each attempt to access pseudo-hashes will trigger the warning `Pseudo-hashes are deprecated`. If you really want to continue using pseudo-hashes but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

Or you can continue to use the *fields* pragma, but please don't expect the data structures to be pseudohashes any more.

(Reminder) 5.005-style threads are deprecated (really)

5.005-style threads (activated by `use Thread;`) were deprecated in Perl 5.8.0 and will be removed after Perl 5.8, see *perl58delta* for details. Each 5.005-style thread creation will trigger the warning `5.005 threads are deprecated`. If you really want to continue using the 5.005 threads but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

(Reminder) The $* variable is deprecated (really)

The `$*` variable controlling multi-line matching has been deprecated and will be removed after 5.8. The variable has been deprecated for a long time, and a deprecation warning `Use of $* is deprecated` is given, now the variable will just finally be removed. The functionality has been supplanted by the `/s` and `/m` modifiers on pattern matching. If you really want to continue using the `$*`-variable but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

Miscellaneous Enhancements

`map` in void context is no longer expensive. `map` is now context aware, and will not construct a list if called in void context.

If a socket gets closed by the server while printing to it, the client now gets a SIGPIPE. While this new feature was not planned, it fell naturally out of PerlIO changes, and is to be considered an accidental feature.

PerlIO::get_layers(FH) returns the names of the PerlIO layers active on a filehandle.

PerlIO::via layers can now have an optional UTF8 method to indicate whether the layer wants to "auto-:utf8" the stream.

utf8::is_utf8() has been added as a quick way to test whether a scalar is encoded internally in UTF-8 (Unicode).

Modules and Pragmata

Updated Modules And Pragmata

The following modules and pragmata have been updated since Perl 5.8.0:

base

B::Bytecode

In much better shape than it used to be. Still far from perfect, but maybe worth a try.

B::Concise

B::Deparse

Benchmark

An optional feature, `:hireswallclock`, now allows for high resolution wall clock times (uses Time::HiRes).

ByteLoader

See B::Bytecode.

bytes

Now has bytes::substr.

CGI

charnames

One can now have custom character name aliases.

CPAN

There is now a simple command line frontend to the CPAN.pm module called *cpan*.

Data::Dumper

A new option, Pair, allows choosing the separator between hash keys and values.

DB_File

Devel::PPPort

Digest::MD5

Encode

Significant updates on the encoding pragma functionality (tr/// and the DATA filehandle, formats).

If a filehandle has been marked as to have an encoding, unmappable characters are detected already during input, not later (when the corrupted data is being used).

The ISO 8859-6 conversion table has been corrected (the 0x30..0x39 erroneously mapped to U+0660..U+0669, instead of U+0030..U+0039). The GSM 03.38 conversion did not handle escape sequences correctly. The UTF-7 encoding has been added (making Encode feature-complete with Unicode::String).

fields

libnet

Math::BigInt

A lot of bugs have been fixed since v1.60, the version included in Perl v5.8.0. Especially noteworthy are the bug in Calc that caused div and mod to fail for some large values, and the fixes to the handling of bad inputs.

Some new features were added, e.g. the broot() method, you can now pass parameters to config() to change some settings at runtime, and it is now possible to trap the creation of NaN and infinity.

As usual, some optimizations took place and made the math overall a tad faster. In some cases, quite a lot faster, actually. Especially alternative libraries like Math::BigInt::GMP benefit from this. In addition, a lot of the quite clunky routines like fsqrt() and flog() are now much much faster.

MIME::Base64

NEXT

Diamond inheritance now works.

Net::Ping

PerlIO::scalar

Reading from non-string scalars (like the special variables, see *perlvar*) now works.

podlators

Pod::LaTeX

PodParsers

Pod::Perldoc

Complete rewrite. As a side-effect, no longer refuses to startup when run by root.

Scalar::Util

New utilities: refaddr, isvstring, looks_like_number, set_prototype.

Storable

Can now store code references (via B::Deparse, so not foolproof).

strict

Earlier versions of the strict pragma did not check the parameters implicitly passed to its "import" (use) and "unimport" (no) routine. This caused the false idiom such as:

```
use strict qw(@ISA);
@ISA = qw(Foo);
```

This however (probably) raised the false expectation that the strict refs, vars and subs were being enforced (and that @ISA was somehow "declared"). But the strict refs, vars, and subs are **not** enforced when using this false idiom.

Starting from Perl 5.8.1, the above **will** cause an error to be raised. This may cause programs which used to execute seemingly correctly without warnings and errors to fail when run under 5.8.1. This happens because

```
use strict qw(@ISA);
```

will now fail with the error:

```
Unknown 'strict' tag(s) '@ISA'
```

The remedy to this problem is to replace this code with the correct idiom:

```
use strict;
use vars qw(@ISA);
@ISA = qw(Foo);
```

Term::ANSIcolor

Test::Harness

Now much more picky about extra or missing output from test scripts.

Test::More

Test::Simple

Text::Balanced

Time::HiRes

Use of nanosleep(), if available, allows mixing subsecond sleeps with alarms.

threads

Several fixes, for example for join() problems and memory leaks. In some platforms (like Linux) that use glibc the minimum memory footprint of one ithread has been reduced by several hundred kilobytes.

threads::shared

Many memory leaks have been fixed.

Unicode::Collate

Unicode::Normalize

Win32::GetFolderPath

Win32::GetOSVersion

Now returns extra information.

Utility Changes

The `h2xs` utility now produces a more modern layout: *Foo-Bar/lib/Foo/Bar.pm* instead of *Foo/Bar/Bar.pm*. Also, the boilerplate test is now called *t/Foo-Bar.t* instead of *t/1.t*.

The Perl debugger (*lib/perl5db.pl*) has now been extensively documented and bugs found while documenting have been fixed.

`perldoc` has been rewritten from scratch to be more robust and feature rich.

`perlcc -B` works now at least somewhat better, while `perlcc -c` is rather more broken. (The Perl compiler suite as a whole continues to be experimental.)

New Documentation

perl573delta has been added to list the differences between the (now quite obsolete) development releases 5.7.2 and 5.7.3.

perl58delta has been added: it is the perldelta of 5.8.0, detailing the differences between 5.6.0 and 5.8.0.

perlartistic has been added: it is the Artistic License in pod format, making it easier for modules to refer to it.

perlcheat has been added: it is a Perl cheat sheet.

perlgpl has been added: it is the GNU General Public License in pod format, making it easier for modules to refer to it.

perlmacosx has been added to tell about the installation and use of Perl in Mac OS X.

perlos400 has been added to tell about the installation and use of Perl in OS/400 PASE.

perlreref has been added: it is a regular expressions quick reference.

Installation and Configuration Improvements

The Unix standard Perl location, */usr/bin/perl*, is no longer overwritten by default if it exists. This change was very prudent because so many Unix vendors already provide a */usr/bin/perl*, but simultaneously many system utilities may depend on that exact version of Perl, so better not to overwrite it.

One can now specify installation directories for site and vendor man and HTML pages, and site and vendor scripts. See *INSTALL*.

One can now specify a destination directory for Perl installation by specifying the DESTDIR variable for `make install`. (This feature is slightly different from the previous `Configure -Dinstallprefix=...`.) See *INSTALL*.

gcc versions 3.x introduced a new warning that caused a lot of noise during Perl compilation: `gcc -Ialreadyknowndirectory (warning: changing search order)`. This warning has now been avoided by Configure weeding out such directories before the compilation.

One can now build subsets of Perl core modules by using the Configure flags `-Dnoextensions=...` and `-Donlyextensions=...`, see *INSTALL*.

Platform-specific enhancements

In Cygwin Perl can now be built with threads (`Configure -Duseithreads`). This works with both Cygwin 1.3.22 and Cygwin 1.5.3.

In newer FreeBSD releases Perl 5.8.0 compilation failed because of trying to use *malloc.h*, which in FreeBSD is just a dummy file, and a fatal error to even try to use. Now *malloc.h* is not used.

Perl is now known to build also in Hitachi HI-UXMPP.

Perl is now known to build again in LynxOS.

Mac OS X now installs with Perl version number embedded in installation directory names for easier upgrading of user-compiled Perl, and the installation directories in general are more standard. In other words, the default installation no longer breaks the Apple-provided Perl. On the other hand, with `Configure -Dprefix=/usr` you can now really replace the Apple-supplied Perl (**please be careful**).

Mac OS X now builds Perl statically by default. This change was done mainly for faster startup times. The Apple-provided Perl is still dynamically linked and shared, and you can enable the sharedness for your own Perl builds by `Configure -Duseshrplib`.

Perl has been ported to IBM's OS/400 PASE environment. The best way to build a Perl for PASE is to use an AIX host as a cross-compilation environment. See README.os400.

Yet another cross-compilation option has been added: now Perl builds on OpenZaurus, an Linux distribution based on Mandrake + Embedix for the Sharp Zaurus PDA. See the Cross/README file.

Tru64 when using gcc 3 drops the optimisation for *toke.c* to `-O2` because of gigantic memory use with the default `-O3`.

Tru64 can now build Perl with the newer Berkeley DBs.

Building Perl on WinCE has been much enhanced, see *README.ce* and *README.perlce*.

Selected Bug Fixes

Closures, eval and lexicals

There have been many fixes in the area of anonymous subs, lexicals and closures. Although this means that Perl is now more "correct", it is possible that some existing code will break that happens to rely on the faulty behaviour. In practice this is unlikely unless your code contains a very complex nesting of anonymous subs, evals and lexicals.

Generic fixes

If an input filehandle is marked `:utf8` and Perl sees illegal UTF-8 coming in when doing `<FH>`, if warnings are enabled a warning is immediately given - instead of being silent about it and Perl being unhappy about the broken data later. (The `:encoding(utf8)` layer also works the same way.)

binmode(SOCKET, ":utf8") only worked on the input side, not on the output side of the socket. Now it works both ways.

For threaded Perls certain system database functions like getpwent() and getgrent() now grow their result buffer dynamically, instead of failing. This means that at sites with lots of users and groups the functions no longer fail by returning only partial results.

Perl 5.8.0 had accidentally broken the capability for users to define their own uppercase<->lowercase Unicode mappings (as advertised by the Camel). This feature has been fixed and is also documented better.

In 5.8.0 this

```
$some_unicode .= <FH>;
```

didn't work correctly but instead corrupted the data. This has now been fixed.

Tied methods like FETCH etc. may now safely access tied values, i.e. resulting in a recursive call to FETCH etc. Remember to break the recursion, though.

At startup Perl blocks the SIGFPE signal away since there isn't much Perl can do about it. Previously this blocking was in effect also for programs executed from within Perl. Now Perl restores the original SIGFPE handling routine, whatever it was, before running external programs.

Linenumbers in Perl scripts may now be greater than 65536, or 2**16. (Perl scripts have always been able to be larger than that, it's just that the linenumber for reported errors and warnings have "wrapped around".) While scripts that large usually indicate a need to rethink your code a bit, such Perl scripts do exist, for example as results from generated code. Now linenumbers can go all the way to 4294967296, or 2**32.

Platform-specific fixes

Linux

- Setting $0 works again (with certain limitations that Perl cannot do much about: see *"$0" in perlvar*)

HP-UX

- Setting $0 now works.

VMS

- Configuration now tests for the presence of `poll()`, and IO::Poll now uses the vendor-supplied function if detected.
- A rare access violation at Perl start-up could occur if the Perl image was installed with privileges or if there was an identifier with the subsystem attribute set in the process's rightslist. Either of these circumstances triggered tainting code that contained a pointer bug. The faulty pointer arithmetic has been fixed.
- The length limit on values (not keys) in the %ENV hash has been raised from 255 bytes to 32640 bytes (except when the PERL_ENV_TABLES setting overrides the default use of logical names for %ENV). If it is necessary to access these long values from outside Perl, be aware that they are implemented using search list logical names that store the value in pieces, each 255-byte piece (up to 128 of them) being an element in the search list. When doing a lookup in %ENV from within Perl, the elements are combined into a single value. The existing VMS-specific ability to access individual elements of a search list logical name via the $ENV{'foo;N'} syntax (where N is the search list index) is unimpaired.
- The piping implementation now uses local rather than global DCL symbols for inter-process communication.
- File::Find could become confused when navigating to a relative directory whose name collided with a logical name. This problem has been corrected by adding directory syntax to relative path names, thus preventing logical name translation.

Win32

- A memory leak in the fork() emulation has been fixed.
- The return value of the ioctl() built-in function was accidentally broken in 5.8.0. This has been corrected.
- The internal message loop executed by perl during blocking operations sometimes interfered with messages that were external to Perl. This often resulted in blocking operations terminating prematurely or returning incorrect results, when Perl was executing under environments that could generate Windows messages. This has been corrected.
- Pipes and sockets are now automatically in binary mode.
- The four-argument form of select() did not preserve $! (errno) properly when there were errors in the underlying call. This is now fixed.
- The "CR CR LF" problem of has been fixed, binmode(FH, ":crlf") is now effectively a no-op.

New or Changed Diagnostics

All the warnings related to pack() and unpack() were made more informative and consistent.

Changed "A thread exited while %d threads were running"

The old version

```
A thread exited while %d other threads were still running
```

was misleading because the "other" included also the thread giving the warning.

Removed "Attempt to clear a restricted hash"

It is not illegal to clear a restricted hash, so the warning was removed.

New "Illegal declaration of anonymous subroutine"

You must specify the block of code for `sub`.

Changed "Invalid range "%s" in transliteration operator"

The old version

```
Invalid [] range "%s" in transliteration operator
```

was simply wrong because there are no "[] ranges" in tr///.

New "Missing control char name in \c"

Self-explanatory.

New "Newline in left-justified string for %s"

The padding spaces would appear after the newline, which is probably not what you had in mind.

New "Possible precedence problem on bitwise %c operator"

If you think this

```
$x & $y == 0
```

tests whether the bitwise AND of $x and $y is zero, you will like this warning.

New "Pseudo-hashes are deprecated"

This warning should have been already in 5.8.0, since they are.

New "read() on %s filehandle %s"

You cannot read() (or sysread()) from a closed or unopened filehandle.

New "5.005 threads are deprecated"

This warning should have been already in 5.8.0, since they are.

New "Tied variable freed while still in use"

Something pulled the plug on a live tied variable, Perl plays safe by bailing out.

New "To%s: illegal mapping '%s'"

An illegal user-defined Unicode casemapping was specified.

New "Use of freed value in iteration"

Something modified the values being iterated over. This is not good.

Changed Internals

These news matter to you only if you either write XS code or like to know about or hack Perl internals (using Devel::Peek or any of the `B::` modules counts), or like to run Perl with the `-D` option.

The embedding examples of *perlembed* have been reviewed to be up to date and consistent: for example, the correct use of PERL_SYS_INIT3() and PERL_SYS_TERM().

Extensive reworking of the pad code (the code responsible for lexical variables) has been conducted by Dave Mitchell.

Extensive work on the v-strings by John Peacock.

UTF-8 length and position cache: to speed up the handling of Unicode (UTF-8) scalars, a cache was introduced. Potential problems exist if an extension bypasses the official APIs and directly modifies the PV of an SV: the UTF-8 cache does not get cleared as it should.

APIs obsoleted in Perl 5.8.0, like sv_2pv, sv_catpvn, sv_catsv, sv_setsv, are again available.

Certain Perl core C APIs like cxinc and regatom are no longer available at all to code outside the Perl core of the Perl core extensions. This is intentional. They never should have been available with the shorter names, and if you application depends on them, you should (be ashamed and) contact perl5-porters to discuss what are the proper APIs.

Certain Perl core C APIs like `Perl_list` are no longer available without their `Perl_` prefix. If your XS module stops working because some functions cannot be found, in many cases a simple fix is to add the `Perl_` prefix to the function and the thread context `aTHX_` as the first argument of the function call. This is also how it should always have been done: letting the Perl_-less forms to leak from the core was an accident. For cleaner embedding you can also force this for all APIs by defining at compile time the cpp define PERL_NO_SHORT_NAMES.

Perl_save_bool() has been added.

Regexp objects (those created with `qr`) now have S-magic rather than R-magic. This fixed regexps of the form /...(??{...;$x})/ to no longer ignore changes made to $x. The S-magic avoids dropping the caching optimization and making (??{...}) constructs obscenely slow (and consequently useless). See also *"Magic Variables" in perlguts*. Regexp::Copy was affected by this change.

The Perl internal debugging macros DEBUG() and DEB() have been renamed to PERL_DEBUG() and PERL_DEB() to avoid namespace conflicts.

`-DL` removed (the leaktest had been broken and unsupported for years, use alternative debugging mallocs or tools like valgrind and Purify).

Verbose modifier `v` added for `-DXv` and `-Dsv`, see *perlrun*.

New Tests

In Perl 5.8.0 there were about 69000 separate tests in about 700 test files, in Perl 5.8.1 there are about 77000 separate tests in about 780 test files. The exact numbers depend on the Perl configuration and on the operating system platform.

Known Problems

The hash randomisation mentioned in *Incompatible Changes* is definitely problematic: it will wake dormant bugs and shake out bad assumptions.

If you want to use mod_perl 2.x with Perl 5.8.1, you will need mod_perl-1.99_10 or higher. Earlier versions of mod_perl 2.x do not work with the randomised hashes. (mod_perl 1.x works fine.) You will also need Apache::Test 1.04 or higher.

Many of the rarer platforms that worked 100% or pretty close to it with perl 5.8.0 have been left a little bit untended since their maintainers have been otherwise busy lately, and therefore there will be more

failures on those platforms. Such platforms include Mac OS Classic, IBM z/OS (and other EBCDIC platforms), and NetWare. The most common Perl platforms (Unix and Unix-like, Microsoft platforms, and VMS) have large enough testing and expert population that they are doing well.

Tied hashes in scalar context

Tied hashes do not currently return anything useful in scalar context, for example when used as boolean tests:

```
if (%tied_hash) { ... }
```

The current nonsensical behaviour is always to return false, regardless of whether the hash is empty or has elements.

The root cause is that there is no interface for the implementors of tied hashes to implement the behaviour of a hash in scalar context.

Net::Ping 450_service and 510_ping_udp failures

The subtests 9 and 18 of lib/Net/Ping/t/450_service.t, and the subtest 2 of lib/Net/Ping/t/510_ping_udp.t might fail if you have an unusual networking setup. For example in the latter case the test is trying to send a UDP ping to the IP address 127.0.0.1.

B::C

The C-generating compiler backend B::C (the frontend being `perlcc -c`) is even more broken than it used to be because of the extensive lexical variable changes. (The good news is that B::Bytecode and ByteLoader are better than they used to be.)

Platform Specific Problems

EBCDIC Platforms

IBM z/OS and other EBCDIC platforms continue to be problematic regarding Unicode support. Many Unicode tests are skipped when they really should be fixed.

Cygwin 1.5 problems

In Cygwin 1.5 the *io/tell* and *op/sysio* tests have failures for some yet unknown reason. In 1.5.5 the threads tests stress_cv, stress_re, and stress_string are failing unless the environment variable PERLIO is set to "perlio" (which makes also the io/tell failure go away).

Perl 5.8.1 does build and work well with Cygwin 1.3: with (uname -a) `CYGWIN_NT-5.0 ... 1.3.22(0.78/3/2) 2003-03-18 09:20 i686 ...` a 100% "make test" was achieved with `Configure -des -Duseithreads`.

HP-UX: HP cc warnings about sendfile and sendpath

With certain HP C compiler releases (e.g. B.11.11.02) you will get many warnings like this (lines wrapped for easier reading):

```
cc: "/usr/include/sys/socket.h", line 504: warning 562:
  Redeclaration of "sendfile" with a different storage class specifier:
    "sendfile" will have internal linkage.
cc: "/usr/include/sys/socket.h", line 505: warning 562:
  Redeclaration of "sendpath" with a different storage class specifier:
    "sendpath" will have internal linkage.
```

The warnings show up both during the build of Perl and during certain lib/ExtUtils tests that invoke the C compiler. The warning, however, is not serious and can be ignored.

IRIX: t/uni/tr_7jis.t falsely failing

The test t/uni/tr_7jis.t is known to report failure under 'make test' or the test harness with certain releases of IRIX (at least IRIX 6.5 and MIPSpro Compilers Version 7.3.1.1m), but if run manually the test fully passes.

Mac OS X: no usemymalloc

The Perl malloc (`-Dusemymalloc`) does not work at all in Mac OS X. This is not that serious, though, since the native malloc works just fine.

Tru64: No threaded builds with GNU cc (gcc)

In the latest Tru64 releases (e.g. v5.1B or later) gcc cannot be used to compile a threaded Perl (-Duseithreads) because the system `<pthread.h>` file doesn't know about gcc.

Win32: sysopen, sysread, syswrite

As of the 5.8.0 release, sysopen()/sysread()/syswrite() do not behave like they used to in 5.6.1 and earlier with respect to "text" mode. These built-ins now always operate in "binary" mode (even if sysopen() was passed the O_TEXT flag, or if binmode() was used on the file handle). Note that this issue should only make a difference for disk files, as sockets and pipes have always been in "binary" mode in the Windows port. As this behavior is currently considered a bug, compatible behavior may be re-introduced in a future release. Until then, the use of sysopen(), sysread() and syswrite() is not supported for "text" mode operations.

Future Directions

The following things **might** happen in future. The first publicly available releases having these characteristics will be the developer releases Perl 5.9.x, culminating in the Perl 5.10.0 release. These are our best guesses at the moment: we reserve the right to rethink.

- PerlIO will become The Default. Currently (in Perl 5.8.x) the stdio library is still used if Perl thinks it can use certain tricks to make stdio go **really** fast. For future releases our goal is to make PerlIO go even faster.
- A new feature called *assertions* will be available. This means that one can have code called assertions sprinkled in the code: usually they are optimised away, but they can be enabled with the `-A` option.
- A new operator `//` (defined-or) will be available. This means that one will be able to say

  ```
  $a // $b
  ```

  instead of

  ```
  defined $a ? $a : $b
  ```

  and

  ```
  $c //= $d;
  ```

  instead of

  ```
  $c = $d unless defined $c;
  ```

  The operator will have the same precedence and associativity as `||`. A source code patch against the Perl 5.8.1 sources will be available in CPAN as *authors/id/H/HM/HMBRAND/dor-5.8.1.diff*.
- `unpack()` will default to unpacking the `$_`.
- Various Copy-On-Write techniques will be investigated in hopes of speeding up Perl.
- CPANPLUS, Inline, and Module::Build will become core modules.

- The ability to write true lexically scoped pragmas will be introduced.
- Work will continue on the bytecompiler and byteloader.
- v-strings as they currently exist are scheduled to be deprecated. The v-less form (1.2.3) will become a "version object" when used with `use`, `require`, and `$VERSION`. $^V will also be a "version object" so the printf("%vd",...) construct will no longer be needed. The v-ful version (v1.2.3) will become obsolete. The equivalence of strings and v-strings (e.g. that currently 5.8.0 is equal to "\5\8\0") will go away. **There may be no deprecation warning for v-strings**, though: it is quite hard to detect when v-strings are being used safely, and when they are not.
- 5.005 Threads Will Be Removed
- The $* Variable Will Be Removed (it was deprecated a long time ago)
- Pseudohashes Will Be Removed

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. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

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.