

NAME

perl572delta - what's new for perl v5.7.2

DESCRIPTION

This document describes differences between the 5.7.1 release and the 5.7.2 release.

(To view the differences between the 5.6.0 release and the 5.7.0 release, see *perl570delta*. To view the differences between the 5.7.0 release and the 5.7.1 release, see *perl571delta*.)

Security Vulnerability Closed

(This change was already made in 5.7.0 but bears repeating here.)

A security vulnerability affecting all Perl versions prior to 5.6.1 was found in August 2000. The vulnerability does not affect default installations and as far as is known affects only the Linux platform.

You should upgrade your Perl to 5.6.1 as soon as possible. Patches for earlier releases exist but using the patches require full recompilation from the source code anyway, so 5.6.1 is your best choice.

See <http://www.cpan.org/src/5.0/sperl-2000-08-05/sperl-2000-08-05.txt> for more information.

Incompatible Changes

64-bit platforms and malloc

If your pointers are 64 bits wide, the Perl malloc is no more being used because it simply does not work with 8-byte pointers. Also, usually the system malloc on such platforms are much better optimized for such large memory models than the Perl malloc.

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 modperl which are using the AIX native interface.

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.

Different Definition of the Unicode Character Classes `\p{In...}`

As suggested by the Unicode consortium, the Unicode character classes now prefer *scripts* as opposed to *blocks* (as defined by Unicode); in Perl, when the `\p{In...}` and the `\P{In...}` regular expression constructs are used. This has changed the definition of some of those character classes.

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 256 characters based on the Unicode numbering.

In general this change results in more inclusive Unicode character classes, but changes to the other direction also do take place: 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`).

Changes in the character class semantics may have happened if a script and a block happen to have the same name, for example `Hebrew`. In such cases the script wins and `\p{InHebrew}` now means the script definition of Hebrew. The block definition is still available, though, by appending `Block` to the name: `\p{InHebrewBlock}` means what `\p{InHebrew}` meant in perl 5.6.0. For the full list of

affected character classes, see *"Blocks" in perlunicode*.

Deprecations

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 syntaxes `@a->[...]` and `@h->{...}` have now been deprecated.

The `suidperl` is also considered to be too much a risk to continue maintaining and the `suidperl` code is likely to be removed in a future release.

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 `chdir(undef)` and `chdir("")` behaviors to match `chdir()` has been deprecated. In future versions, `chdir(undef)` and `chdir("")` will simply fail.

Core Enhancements

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.

- The rules for allowing underscores (underbars) in numeric constants have been relaxed and simplified: now you can have an underscore **between digits**.
- GMAGIC (right-hand side magic) could in many cases such as string concatenation be invoked too many times.
- 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.
- Lvalue subroutines can now return `undef` in list context.
- The `op_clear` and `op_null` are now exported.
- A new special regular expression variable has been introduced: `$$^N`, which contains the most-recently closed group (submatch).
- `utime` now supports `utime undef, undef, @files` to change the file timestamps to the current time.
- The Perl parser has been stress tested using both random input and Markov chain input.
- `eval "v200"` now works.
- VMS now works under PerlIO.
- END blocks are now run even if you `exit/die` in a BEGIN block. 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*.

Modules and Pragmata

New Modules and Distributions

- *Attribute::Handlers* - Simpler definition of attribute handlers
- *ExtUtils::Constant* - generate XS code to import C header constants
- *I18N::Langinfo* - query locale information
- *I18N::LangTags* - functions for dealing with RFC3066-style language tags
- *libnet* - a collection of perl5 modules related to network programming
Perl installation leaves libnet unconfigured, use *libnetcfg* to configure.
- *List::Util* - selection of general-utility list subroutines
- *Locale::Maketext* - framework for localization
- *Memoize* - Make your functions faster by trading space for time
- *NEXT* - pseudo-class for method redispach
- *Scalar::Util* - selection of general-utility scalar subroutines
- *Test::More* - yet another framework for writing test scripts
- *Test::Simple* - Basic utilities for writing tests
- *Time::HiRes* - high resolution ualarm, usleep, and gettimeofday
- *Time::Piece* - Object Oriented time objects
(Previously known as *Time::Object*.)
- *Time::Seconds* - a simple API to convert seconds to other date values
- *UnicodeCD* - Unicode Character Database

Updated And Improved Modules and Pragmata

- *B::Deparse* module has been significantly enhanced. It now can 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.
- *Class::Struct* now assigns the array/hash element if the accessor is called with an array/hash element as the **sole** argument.
- *Cwd* extension is now (even) faster.
- *DB_File* extension has been updated to version 1.77.
- *Fcntl*, *Socket*, and *Sys::Syslog* have been rewritten to use the new-style constant dispatch section (see *ExtUtils::Constant*).
- *File::Find* is now (again) reentrant. It also has been made more portable.
- *File::Glob* now supports `GLOB_LIMIT` constant to limit the size of the returned list of filenames.
- *IO::Socket::INET* now supports `LocalPort` of zero (usually meaning that the operating system will make one up.)
- The *vars* pragma now supports declaring fully qualified variables. (Something that `our()` does not and will not support.)

Utility Changes

- The *emacs/e2ctags.pl* is now much faster.
- *h2ph* now supports C trigraphs.
- *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** gets 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 the libnet.
- The *Pod::Html* (and thusly *pod2html*) now allows specifying a cache directory.

New Documentation

- *Locale::Maketext::TPJ13* is an article about software localization, originally published in The Perl Journal #13, republished here with kind permission.
- More README.\$PLATFORM files have been converted into pod, which also means that they also be installed as perl\$PLATFORM documentation files. The new files are *perlapollo*, *perlbeos*, *perldgux*, *perlhurd*, *perlmint*, *perlnetware*, *perlplan9*, *perlqnx*, and *perltru64*.
- The *Todo* and *Todo-5.6* files have been merged into *perltodo*.
- Use of the *gprof* tool to profile Perl has been documented in *perlhack*. There is a make target "perl.gprof" for generating a gprofiled Perl executable.

Installation and Configuration Improvements

New Or Improved Platforms

- 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.
- DG/UX platform now supports the 5.005-style threads. See *perldgux*.
- DYNIX/ptx platform (a.k.a. dynixptx) is supported at or near osvers 4.5.2.
- Several Mac OS (Classic) portability patches have been applied. We hope to get a fully working port by 5.8.0. (The remaining problems relate to the changed IO model of Perl.) See *perlmacos*.
- Mac OS X (or Darwin) should now be able to build Perl even on HFS+ filesystems. (The case-insensitivity confused the Perl build process.)
- NetWare from Novell is now supported. See *perlnetware*.
- The Amdahl UTS UNIX mainframe platform is now supported.

Generic Improvements

- 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`.
- 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.

- 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`).
- The `B::Deparse` compiler backend has been so significantly improved that almost the whole Perl test suite passes after being deparsed. A make target has been added to help in further testing: `make test.deparse`.

Selected Bug Fixes

- The `autouse` pragma didn't work for `Multi::Part::Function::Names`.
- 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 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.
- `dprofp -R` didn't work.
- `PERL5OPT` with embedded spaces didn't work.
- `Sys::Syslog` ignored the `LOG_AUTH` constant.

Platform Specific Changes and Fixes

- 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.

New or Changed Diagnostics

- In the regular expression diagnostics the `<< HERE` marker introduced in 5.7.0 has been changed to be `<<- HERE` since too many people found the `<<` to be too similar to here-document starters.
- If you try to *"pack" in `perfunc`* 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.
- 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.
- Using arrays or hashes as references (e.g. `%foo->{bar}`) has been deprecated for a while. Now you will get an optional warning.

Source Code Enhancements

MAGIC constants

The MAGIC constants (e.g. `'P'`) have been macrofied (e.g. `PERL_MAGIC_TIED`) for better source code readability and maintainability.

Better commented code

`perly.c`, `sv.c`, and `sv.h` have now been extensively commented.

Regex pre-/post-compilation items matched up

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 *perldebug* for more complete information.

gcc -Wall

The C code has been made much more `gcc -Wall` clean. Some warning messages still remain, though, so if you are compiling with `gcc` you will see some warnings about dubious practices. The warnings are being worked on.

New Tests

Several new tests have been added, especially for the *lib* subsection.

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

Note that unlike other sections in this document (which describe changes since 5.7.0) this section is cumulative containing known problems for all the 5.7 releases.

AIX

- 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 the `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 few random tests failing, 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. "`!slpp -L|grep vac.C`" will tell you the `vac` version.

Amiga Perl Invoking Mystery

One cannot call Perl using the `volume:` syntax, that is, `perl -v` works, but for example `bin:perl -v` doesn't. The exact reason is known but the current suspect is the *ixemul* library.

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

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

Cygwin intermittent failures of lib/Memoize/t/expire_file 11 and 12

The subtests 11 and 12 sometimes fail and sometimes work.

HP-UX lib/io_multihomed Fails When LP64-Configured

The `lib/io_multihomed` test may hang in HP-UX if Perl has been configured to be 64-bit. Because other 64-bit platforms do not hang in this test, HP-UX is suspect. All other tests pass in 64-bit HP-UX. The test attempts to create and connect to "multihomed" sockets (sockets which have multiple IP addresses).

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 Sflo Fails op/misc Test 48

No known fix.

OS/390

OS/390 has rather many test failures but the situation is actually 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/B/Deparse.t			14	1	7.14%	14
../ext/B/Showlex.t			1	1	100.00%	1
../ext/Encode/Encode/Tcl.t			610	13	2.13%	592 594 596
598						
						600 602
604-610						
../ext/IO/lib/IO/t/io_unix.t	113	28928	5	3	60.00%	3-5
../ext/POSIX/POSIX.t			29	1	3.45%	14
../ext/Storable/t/lock.t	255	65280	5	3	60.00%	3-5
../lib/locale.t	129	33024	117	19	16.24%	99-117
../lib/warnings.t			434	1	0.23%	75
../lib/ExtUtils.t			27	1	3.70%	25
../lib/Math/BigInt/t/bigintpm.t			1190	1	0.08%	1145
../lib/Unicode/UCD.t			81	48	59.26%	1-16 49-64
66-81						
../lib/User/pwent.t			9	1	11.11%	4
op/pat.t			660	6	0.91%	242-243
424-425						
						626-627
op/split.t	0	9	??	??	%	??
op/taint.t			174	3	1.72%	156 162 168
op/tr.t			70	3	4.29%	50 58-59
Failed 16/422 test scripts, 96.21% okay. 105/23251 subtests failed, 99.55% okay.						

op/sprintf tests 129 and 130

The op/sprintf tests 129 and 130 are known to fail on some platforms. Examples include any platform using sfio, and Compaq/Tandem's NonStop-UX. The failing platforms do not comply with the ANSI C Standard, line 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".)

Failure of Thread tests

Note that support for 5.005-style threading remains experimental.

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.

lib/autouse.t	4
t/lib/thr5005.t	19-20

UNICOS

- ext/POSIX/sigaction subtests 6 and 13 may fail.
- lib/ExtUtils may spuriously claim that subtest 28 failed, which is interesting since the test only has 27 tests.
- Numerous numerical test failures

op/numconvert	209, 210, 217, 218
op/override	7

ext/Time/HiRes/HiRes	9
lib/Math/BigInt/t/bigintpm	1145
lib/Math/Trig	25

These tests fail because of yet unresolved floating point inaccuracies.

UTS

There are a few known test failures, see *perluts*.

VMS

Rather many tests are failing in VMS but that actually more tests succeed in VMS than they used to, it's just that there are many, many more tests than there used to be.

Here are the known failures from some compiler/platform combinations.

DEC C V5.3-006 on OpenVMS VAX V6.2

```
[-.ext.list.util.t]tainted.....FAILED on test 3
[-.ext.posix]sigaction.....FAILED on test 7
[-.ext.time.hires]hires.....FAILED on test 14
[-.lib.file.find]taint.....FAILED on test 17
[-.lib.math.bigint.t]bigintpm.....FAILED on test 1183
[-.lib.test.simple.t]exit.....FAILED on test 1
[.lib]vmsish.....FAILED on test 13
[.op]sprintf.....FAILED on test 12
Failed 8/399 tests, 91.23% okay.
```

DEC C V6.0-001 on OpenVMS Alpha V7.2-1 and Compaq C V6.2-008 on OpenVMS Alpha V7.1

```
[-.ext.list.util.t]tainted.....FAILED on test 3
[-.lib.file.find]taint.....FAILED on test 17
[-.lib.test.simple.t]exit.....FAILED on test 1
[.lib]vmsish.....FAILED on test 13
Failed 4/399 tests, 92.48% okay.
```

Compaq C V6.4-005 on OpenVMS Alpha 7.2.1

```
[-.ext.b]showlex.....FAILED on test 1
[-.ext.list.util.t]tainted.....FAILED on test 3
[-.lib.file.find]taint.....FAILED on test 17
[-.lib.test.simple.t]exit.....FAILED on test 1
[.lib]vmsish.....FAILED on test 13
[.op]misc.....FAILED on test 49
Failed 6/401 tests, 92.77% okay.
```

Win32

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

Localising a Tied Variable Leaks Memory

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

...

local($tie_hash{Foo}) = 1; # leaks
```

Code like the above is known to leak memory every time the `local()` is executed.

Self-tying of Arrays and Hashes Is Forbidden

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 for now forbidden (you will get a fatal error even from an attempt).

Variable Attributes are not Currently Usable for Tying

This limitation will hopefully be fixed in future. (Subroutine attributes work fine for tying, see *Attribute::Handlers*).

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 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 largefile-ness. 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) link together at all binaries with different ideas about file offsets, all this is platform-dependent.

The Compiler Suite Is Still Experimental

The compiler suite is slowly getting better but is nowhere near working order yet.

The Long Double Support is Still Experimental

The ability to configure Perl's numbers to use "long doubles", floating point numbers of hopefully better accuracy, is still experimental. The implementations of long doubles are not yet widespread and the existing implementations are not quite mature or standardised, therefore trying to support them is a rare and moving target. The gain of more precision may also be offset by slowdown in computations (more bits to move around, and the operations are more likely to be executed by less optimised libraries).

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/perl/>, 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>, with many contributions from The Perl Porters and Perl Users submitting feedback and patches.

Send omissions or corrections to <perlbug@perl.org>.