NAME

perl571delta - what's new for perl v5.7.1

DESCRIPTION

This document describes differences between the 5.7.0 release and the 5.7.1 release.

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

Security Vulnerability Closed

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

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 April 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 all the Perl 5.7 releases (and will be gone also from the maintenance release 5.6.1), so that particular vulnerability isn't there anymore. However, further security vulnerabilities are, unfortunately, always possible. The suidperl code is being reviewed and if deemed too risky to continue to be supported, it may be completely removed from future releases. 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/).

Incompatible Changes

- 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*.
- The list of filenames from glob() (or <...>) is now by default sorted alphabetically to be csh-compliant. (bsd_glob() does still sort platform natively, ASCII or EBCDIC, unless GLOB_ALPHASORT is specified.)

Core Enhancements

AUTOLOAD Is Now Lvaluable

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

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.

- 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.

- 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.

- The list form of `open` is now implemented for pipes (at least on UNIX):

  ```
  open($fh,"-|", 'cat', '/etc/motd')
  ```

  creates a pipe, and runs the equivalent of exec('cat', '/etc/motd') in the child process.

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

- Formats now support zero-padded decimal fields.

- 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 into often slightly faster and always less lossy arithmetics. (Previously Perl always preferred floating point numbers in its math.)

- The printf() and sprintf() now support parameter reordering using the `%\d+\$` and `*\d+\$` syntaxes. For example

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

  will print "bar foo\n"; This feature helps in writing internationalised software.

- Unicode in general should be now much more usable. Unicode can be used in hash keys, Unicode in regular expressions should work now, Unicode in tr/// should work now (though tr/// seems to be a particularly tricky to get right, so you have been warned)

- The Unicode Character Database coming with Perl has been upgraded to Unicode 3.1. For more information, see http://www.unicode.org/ , and http://www.unicode.org/unicode/reports/tr27/

  For developers interested in enhancing Perl's Unicode capabilities: almost all the UCD files are included with the Perl distribution in the lib/unicode subdirectory. The most notable omission, for space considerations, is the Unihan database.

- The Unicode character classes \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.)

Signals Are Now Safe

Perl used to be fragile in that signals arriving at inopportune moments could corrupt Perl's internal state.

Modules and Pragmata

New Modules

- 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* for more information.

- Class::ISA, by Sean Burke, for reporting the search path for a class's ISA tree, has been added.

  See *Class::ISA* for more information.

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

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

  See *Digest* for more information.

- Digest::MD5 for calculating MD5 digests (checksums), by Gisle Aas, has been added.

```
    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 use is discouraged.

  See *Digest::MD5* for more information.

- Encode, by Nick Ing-Simmons, provides a mechanism to translate between different character encodings. Support for Unicode, ISO-8859-*, ASCII, CP*, KOI8-R, and three variants of EBCDIC are compiled in to the module. Several other encodings (like Japanese, Chinese, and MacIntosh encodings) are included and will be loaded at runtime.

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

  See *Encode* for more information.

- Filter::Simple is an easy-to-use frontend to Filter::Util::Call, from Damian Conway.

```
    # 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"
```

See *Filter::Simple* for more information.

- 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* for more information.
- Locale::Constants, Locale::Country, Locale::Currency, and Locale::Language, from Neil Bowers, have been added. They provide the codes for various locale standards, such as "fr" for France, "usd" for US Dollar, and "jp" 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* for more information.

- MIME::Base64, by Gisle Aas, allows you to encode data in base64.

```
    use MIME::Base64;

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

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

See *MIME::Base64* for more information.

- MIME::QuotedPrint, by Gisle Aas, allows you to encode data in quoted-printable encoding.

```
    use MIME::QuotedPrint;

    $encoded = encode_qp("Smiley in Unicode: \x{263a}");
    $decoded = decode_qp($encoded);

    print $encoded, "\n"; # "Smiley in Unicode: =263A"
```

MIME::QuotedPrint has been enhanced to provide the basic methods necessary to use it with PerlIO::Via as in :

```
    use MIME::QuotedPrint;
    open($fh,">Via(MIME::QuotedPrint)",$path)
```

See *MIME::QuotedPrint* for more information.

- 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 layer. Other future possibilities include PerlIO::Array and PerlIO::Code. See *PerlIO::Scalar* for more information.
- 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).

```
    use MIME::QuotedPrint;
    open($fh,">Via(MIME::QuotedPrint)",$path)
```

  This will automatically convert everything output to `$fh` to Quoted-Printable. See *PerlIO::Via* for more information.

- Pod::Text::Overstrike, by Joe Smith, has been added. It converts POD data to formatted overstrike text. See *Pod::Text::Overstrike* for more information.
- Switch from 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* for more information.

- Text::Balanced from 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* for more information.

- Tie::RefHash::Nestable, by Edward Avis, allows storing hash references (unlike the standard Tie::RefHash) The module is contained within Tie::RefHash.

- XS::Typemap, by Tim Jenness, is a test extension that exercises XS typemaps. Nothing gets installed but for extension writers the code is worth studying.

Updated And Improved Modules and Pragmata

- B::Deparse should be now more robust. It still far from providing a full round trip for any random piece of Perl code, though, and is under active development: expect more robustness in 5.7.2.
- Class::Struct can now define the classes in compile time.
- Math::BigFloat has undergone much fixing, and in addition the fmod() function now supports modulus operations.

  (The fixed Math::BigFloat module is also available in CPAN for those who can't upgrade their Perl: http://www.cpan.org/authors/id/J/JP/JPEACOCK/)
- 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).
- IO::Socket has now 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 has support for ReusePort option (if your platform supports it). The Reuse option now has an alias, ReuseAddr. For clarity you may want to prefer ReuseAddr.
- Net::Ping has been enhanced. There is now "external" protocol which uses Net::Ping::External module which runs external ping(1) and parses the output. An alpha version of Net::Ping::External is available in CPAN and in 5.7.2 the Net::Ping::External may be integrated to Perl.
- The `open` pragma allows layers other than ":raw" and ":crlf" when using PerlIO.
- 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.
- The Test module has been significantly enhanced. Its use is greatly recommended for module writers.
- 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.

The following modules have been upgraded from the versions at CPAN: CPAN, CGI, DB_File, File::Temp, Getopt::Long, Pod::Man, Pod::Text, Storable, Text-Tabs+Wrap.

Performance Enhancements

- 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.

Utility Changes

- h2xs now produces template README.
- s2p has been completely rewritten in Perl. (It is in fact a full implementation of sed in Perl.)
- xsubpp now supports OUT keyword.

New Documentation

perlclib

Internal replacements for standard C library functions. (Interesting only for extension writers and Perl core hackers.)

perliol

Internals of PerlIO with layers.

README.aix

Documentation on compiling Perl on AIX has been added. AIX has several different C compilers and getting the right patch level is essential. On install README.aix will be installed as *perlaix*.

README.bs2000

Documentation on compiling Perl on the POSIX-BC platform (an EBCDIC mainframe environment) has been added.

This was formerly known as README.posix-bc but the name was considered to be too confusing (it has nothing to do with the POSIX module or the POSIX standard). On install README.bs2000 will be installed as *perlbs2000*.

README.macos

In perl 5.7.1 (and in the 5.6.1) the MacPerl sources have been synchronised with the standard Perl sources. To compile MacPerl some additional steps are required, and this file documents those steps. On install README.macos will be installed as *perlmacos*.

README.mpeix

The README.mpeix has been podified, which means that this information about compiling and using Perl on the MPE/iX miniframe platform will be installed as *perlmpeix*.

README.solaris

README.solaris has been created and Solaris wisdom from elsewhere in the Perl documentation has been collected there. On install README.solaris will be installed as *perlsolaris*.

README.vos

The README.vos has been podified, which means that this information about compiling and using Perl on the Stratus VOS miniframe platform will be installed as *perlvos*.

Porting/repository.pod

Documentation on how to use the Perl source repository has been added.

Installation and Configuration Improvements

- 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.)
- APPLLIB_EXP, a less-know 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.
- Building Berkeley DB3 for compatibility modes for DB, NDBM, and ODBM has been documented in INSTALL.
- 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.

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.
- After a long pause, AmigaOS has been verified to be happy with Perl.
- 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.
- 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.
- Mac OS Classic (MacPerl has of course been available since perl 5.004 but now the source code bases of standard Perl and MacPerl have been synchronised)
- NCR MP-RAS is now supported.
- NonStop-UX is now supported.
- Amdahl UTS is now supported.
- z/OS (formerly known as OS/390, formerly known as MVS OE) has now support for dynamic loading. This is not selected by default, however, you must specify -Dusedl in the arguments of Configure.

Generic Improvements

- 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.
- Some new Configure symbols, useful for extension writers:

  d_cmsghdr

  For struct cmsghdr.

  d_fcntl_can_lock

  Whether fcntl() can be used for file locking.

  d_fsync

  d_getitimer

  d_getpagsz

  For getpagesize(), though you should prefer POSIX::sysconf(_SC_PAGE_SIZE))

  d_msghdr_s

  For struct msghdr.

  need_va_copy

  Whether one needs to use Perl_va_copy() to copy varargs.

  d_readv

  d_recvmsg

  d_sendmsg

sig_size

The number of elements in an array needed to hold all the available signals.

d_sockatmark

d_strtoq

d_u32align

Whether one needs to access character data aligned by U32 sized pointers.

d_ualarm

d_usleep

- Removed Configure symbols: the PDP-11 memory model settings: huge, large, medium, models.
- SOCKS support is now much more robust.
- 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.

Selected Bug Fixes

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

- chop(@list) in list context returned the characters chopped in reverse order. This has been reversed to be in the right order.
- The order of DESTROYs has been made more predictable.
- mkdir() now ignores trailing slashes in the directory name, as mandated by POSIX.
- Attributes (like :shared) didn't work with our().
- The PERL5OPT environment variable (for passing command line arguments to Perl) didn't work for more than a single group of options.
- 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.
- All but the first argument of the IO syswrite() method are now optional.
- Tie::ARRAY SPLICE method was broken.
- vec() now tries to work with characters <= 255 when possible, but it leaves higher character values in place. In that case, if vec() was used to modify the string, it is no longer considered to be utf8-encoded.

Platform Specific Changes and Fixes

- Linux previously had problems related to sockaddrlen when using accept(), revcfrom() (in Perl: recv()), getpeername(), and getsockname().

- Previously DYNIX/ptx had problems in its Configure probe for non-blocking I/O.
- Windows
  - 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++).
  - Win32::GetCwd() correctly returns C:\ instead of C: when at the drive root. Other bugs in chdir() and Cwd::cwd() have also been fixed.
  - Duping socket handles with open(F, ">&MYSOCK") now works under Windows 9x.
  - HTML files will be installed in c:\perl\html instead of c:\perl\lib\pod\html
  - The makefiles now provide a single switch to bulk-enable all the features enabled in ActiveState ActivePerl (a popular binary distribution).

New or Changed Diagnostics

Two new debugging options have been added: if you have compiled your Perl with debugging, you can use the -DT and -DR options to trace tokenising and to add reference counts to displaying variables, respectively.

- 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 evaled code.

Changed Internals

- Some new APIs: ptr_table_clear(), ptr_table_free(), sv_setref_uv(). For the full list of the available APIs see *perlapi*.
- dTHR and djSP have been obsoleted; the former removed (because it's a no-op) and the latter replaced with dSP.
- Perl now uses system malloc instead of Perl malloc on all 64-bit platforms, and even in some not-always-64-bit platforms like AIX, IRIX, and Solaris. This change breaks backward compatibility but Perl's malloc has problems with large address spaces and also the speed of vendors' malloc is generally better in large address space machines (Perl's malloc is mostly tuned for space).

New Tests

Many new tests have been added. The most notable is probably the lib/1_compile: it is very notable because running it takes quite a long time -- it test compiles all the Perl modules in the distribution. Please be patient.

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 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. "lslpp -L|grep vac.C" will tell you the vac version.

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

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

lib/io_multihomed Fails In LP64-Configured HP-UX

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).

Test lib/posix Subtest 9 Fails In LP64-Configured HP-UX

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.

lib/b test 19

The test fails on various platforms (PA64 and IA64 are known), but the exact cause is still being investigated.

Linux With Sfio Fails op/misc Test 48

No known fix.

sigaction test 13 in VMS

The test is known to fail; whether it's because of VMS of because of faulty test is not known.

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 else 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

The subtests 19 and 20 of lib/thr5005.t test 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. (Note that support for 5.005-style threading remains experimental.)

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).

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 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) 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.

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*>.