What's new in Perl 5.8.0?

Paul Fenwick

<pjf@perltraining.com.au>

8th September, 2002

Why move to Perl 5.8.0?

So, you've noticed that Perl 5.8.0 has been released, and you're wondering what it has over Perl 5.6.1? Maybe you've heard a little about this PerlIO thing, and want to know more about it. Well, this talk is for you.

This talk covers some of the new features, changes, and pitfalls associated with Perl 5.8.0. Because I'm lazy, most things will be spoken about in the order they appear in `perldelta`.

Binary Incompatibility

- Perl has a new way of handling I/O, called PerlIO, we'll hear more about that later.

- Under the hood, everything needs to use the new PerlIO hooks and whistles. As such, Perl 5.8.0 is not binary compatible with previous releases of Perl. Any XS modules *must* be re-compiled.

- Pure Perl modules will continue to work.

More testing.

From the "Highlights in 5.8.0" mentioned in the release notes:

```
Perl has now almost six times as many tests
as in 5.6, and the code is test built daily
on several platforms.
```

In short, re-compiling your XS modules aside, Perl 5.8.0 should work straight out of the box.

New ways of opening temporary files

Ever needed a filehandle for something, but didn't really want to use a *real* file? Well, you can now open files in-memory:

```
open($fh, '>', \$variable) or die "...";
```

Anonymous temporary files are easier than every before. No modules to use, or `POSIX`isms to figure out. Now you can just do this:

```
open($fh, "+>", undef) or die "...";
```

Filehandle layers and PerlIO

- What's so nifty about PerlIO? Well, one of the best things are support for filehandle *layers*.

- Layers perform seamless transformations to data at the input/output level. Think about `CRLF` conversion on Windows systems. Layers are like that, only *much* cooler.

- Layers used to be called disciplines in the third-edition camel book. They're now called layers so we can rejoice in a fourth-edition of the camel.

- Let's see a layer in action. How about converting line-feeds to `CRLF`, and writing out a valid UTF-8 stream? That's easy:

```
open($fh, ">:crlf :utf8", $file) or die "...";
```

Layers continued.

- Let's read a locale-specific encoding and convert it into Perl's internal unicode. Easy!

  ```
  # It's all Greek to me!
  open($fh, "<:encoding(iso-8859-7)",$file)
          or die "...";
  ```

- How about some binary data?

  ```
  open($fh, "<:raw",$file) or die "...";
  ```

- Yes, `binmode` still works, and is encouraged even on systems where you didn't think you needed it. It's backwards compatible, and means you'll never have Perl thinking your streams are UTF-8 when you upgrade.

Much better Unicode support.

- As you've probably guessed from our discussion about layers, Perl has better support for Unicode and locales.

- Unicode 3.2.0 support.

- Unicodeness is now bound to data, not lexically scoped. This is much more sensible. Concatenating a unicode string to something results in another unicode string, regardless of `use utf8;`.

- `use utf8;` is now only needed to indicate that your script is written in UTF-8.

No more pseudo-hashes.

- Pseudo-hashes are deprecated, and will be removed in Perl 5.10.0. This is because while pseudohashes were really fast, they made normal-hashes and arrays really slow.

- `Hash::Util` provides all the coolness of the old pseudo-hashes with new "restricted hashes". This can be used to restrict the keys that can be used in a hash, and even make some values read-only.

- `Class::PseudoHash` provides a way of using pseudo-hash style internals in a forward-compatible manner.

New threads.

- The old 5.005 threads, where data-sharing was implicit, is deprecated.
- The new interpreter threads (ithreads) is in. Data-sharing between threads now needs to be declared explicitly.
- Lots of new and/or improved modules and attributes to make it easier to work with threads.

OS Specific changes

- Binary compatibility is even more broken under AIX, because shared libraries now loaded differently as well. You *really must* recompile your XS modules under AIX.

- Perl 5.8.0 does not compile on the Amiga. Something broke during threads development, and a good Amiga expert is needed to help debug the problem.

- Perl 5.8.0 is now supported on AtheOS, BeOS, DYNIX, various EBCDIC systems, MacOS-X, NetWare, and SUPER-UX.

OS changes continued.

- Perl has better memory management on 64-bit systems.

- Socket extensions under VMS are now dynamically loaded instead of statically compiled. This may be a problem with old TCP/IP stacks, but we don't know since nobody uses VMS.

New gotchas!

- A reference to a reference now stringifies as `REF(0x9378fea3)` instead of the old `SCALAR(0x9378fea3)`. This is now consistent with the values returned by `ref()`.

- Perl's internal hash-key order has changed. Nobody depended on it anyway. Oh, except `Data::Dumper`, but nobody used that.

Things you thought already worked, actually do!

- `glob` or the globbing diamond operator now return files sorted alphabetically. This is exactly the same as it always has been on UNIX platforms, and how it was documented in many places.

- Signals can no longer destroy Perl's internal state. Perl now postpones signals until the end of op-codes. This means signals might take a little slower to be processed.

- Signals will still interrupt (potentially) blocking operations.

- Perl has more accurate handling of floating point numbers, and will try to store numbers internally as integers whenever possible. This makes things faster, too.

Things you never used are gone.

- You can no-longer use upper-case string comparison operators (EQ, NE, LT, etc).
- Prototypes such as `sub foo (@bar)` which may indicate you have no idea how prototypes work will generate a warning. This may turn into a fatal error in future releases.
- `suidperl` is considered far too complex to be considered secure. Please don't pretend that it is.
- Nobody knows what `bless(REF, REF)` should do, so don't do that.

Other things that are gone.

- `dump()` is considered deprecated, so use `CORE::dump()` instead. Getting Perl to dump core on demand has probably outlived its usefulness anyway.

- `lstat(FILEHANDLE)` never made any sense, so it now gives a warning, and still makes no sense.

Null package is gay.

- Entering the null package (the statement "`package;`" with no arguments) is now deprecated. Nobody knew what it was supposed to do anyway.

- If you previously used the null package as a perverse way of requiring variables to be qualified, then take a look at `use strict`.

- If you previously used the null package as a perverse form of job security, then consider overriding the `readline` built-in function.

New esoteric bits.

- `AUTOLOAD` is now lvauable.
- `chop()`, `chomp()`, and `readline()` can be overridden.
- The new special variable `$^N` contains the contents of the most recently closed grouping (ie, the group with the rightmost closing parenthesis) for the last successful search pattern.
- `pack('U0a*',...)` can be used to force a string to UTF-8.

New security changes.

- `-t` allows you to run in taint-mode with warnings rather than errors generated by taint violations. Obviously this is for debugging only.

- `exec LIST` and `system LIST` will generate warnings when used with tainted arguments. In the future they'll generate errors.

- The special `${^TAINT}` variable indicates if taint mode is enabled.

- `Scalar::Util` provides a `tainted()` subroutine to check for tainted data.

- The return value of `Cwd::fastcwd()` is considered tainted.

- `%ENV` is tainted correctly under VMS.

Known problems

- The compiler suite is still experimental. That's not really a change.

- Nobody's really sure what `local`ising a tied array or hash should do. So don't do that.

- You can modify `$_` when it's an alias for a non-lvalue in a `for` loop. However this doesn't appear to actually do anything.

- Self-tying of arrays and hashes is broken, so don't do that. You can self-type scalars and IO-thingies as much as you want. You can self-tie globs as long as you remember to explicitly untie at the end.

- Tied or magical array and hash elements don't autovivify.

New modules and pragmata.

Perl 5.8.0 has more standard modules than every before, including everyone's favourite `NEXT` module. Many existing modules have been upgraded an improved.

There are far too many to list here.

Perl also has new pragmata, including **`bignum`**, **`bigint`**, **`bigrat`**, **`if`**, **`open`**, **`sort`**, **`threads`**, and **`threads::shared`**.

In conclusion

- Perl 5.8.0 has great support for Unicode.
- Perl 5.8.0 has lots of neat IO layers.
- Perl 5.8.0 has good threads support.
- Perl 5.8.0 works pretty much everywhere.
- Perl 5.8.0 requires you to recompile *all* your XS modules. Ick.
- Perl 5.8.0 is better, faster, stronger, and more robust.