

NAME

perltodo - Perl TO-DO List

DESCRIPTION

This is a list of wishes for Perl. Send updates to *perl5-porters@perl.org*. If you want to work on any of these projects, be sure to check the perl5-porters archives for past ideas, flames, and propaganda. This will save you time and also prevent you from implementing something that Larry has already vetoed. One set of archives may be found at:

<http://www.xray.mpe.mpg.de/mailling-lists/perl5-porters/>

assertions

Clean up and finish support for assertions. See *assertions*.

iCOW

Sarathy and Arthur have a proposal for an improved Copy On Write which specifically will be able to COW new ithreads. If this can be implemented it would be a good thing.

(?{...}) closures in regexps

Fix (or rewrite) the implementation of the `/ (? { . . . }) /` closures.

A re-entrant regexp engine

This will allow the use of a regex from inside `(?{ })`, `(??{ })` and `(?(?{ })|)` constructs.

pragmata

lexical pragmas

Reimplement the mechanism of lexical pragmas to be more extensible. Fix current pragmas that don't work well (or at all) with lexical scopes or in run-time `eval(STRING)` (`sort`, `re`, `encoding` for example). MJD has a preliminary patch that implements this.

use less 'memory'

Investigate trade offs to switch out perl's choices on memory usage. Particularly perl should be able to give memory back.

prototypes and functions

_ prototype character

Study the possibility of adding a new prototype character, `_`, meaning "this argument defaults to `$_`".

inlining autoloaded constants

Currently the optimiser can inline constants when expressed as subroutines with prototype (`$`) that return a constant. Likewise, many packages wrapping C libraries export lots of constants as subroutines which are AUTOLOADED on demand. However, these have no prototypes, so can't be seen as constants by the optimiser. Some way of cheaply (low syntax, low memory overhead) to the perl compiler that a name is a constant would be great, so that it knows to call the AUTOLOAD routine at compile time, and then inline the constant.

Finish off lvalue functions

The old perltodo notes "They don't work in the debugger, and they don't work for list or hash slices."

Unicode and UTF8

Implicit Latin 1 => Unicode translation

Conversions from byte strings to UTF-8 currently map high bit characters to Unicode without translation (or, depending on how you look at it, by implicitly assuming that the byte strings are in Latin-1). As perl assumes the C locale by default, upgrading a string to UTF-8 may change the meaning of its contents regarding character classes, case mapping, etc. This should probably emit a warning (at least).

UTF8 caching code

The string position/offset cache is not optional. It should be.

Unicode in Filenames

chdir, chmod, chown, chroot, exec, glob, link, lstat, mkdir, open, opendir, qx, readdir, readlink, rename, rmdir, stat, symlink, sysopen, system, truncate, unlink, utime, -X. All these could potentially accept Unicode filenames either as input or output (and in the case of system and qx Unicode in general, as input or output to/from the shell). Whether a filesystem - an operating system pair understands Unicode in filenames varies.

Known combinations that have some level of understanding include Microsoft NTFS, Apple HFS+ (In Mac OS 9 and X) and Apple UFS (in Mac OS X), NFS v4 is rumored to be Unicode, and of course Plan 9. How to create Unicode filenames, what forms of Unicode are accepted and used (UCS-2, UTF-16, UTF-8), what (if any) is the normalization form used, and so on, varies. Finding the right level of interfacing to Perl requires some thought. Remember that an OS does not implicate a filesystem.

(The Windows -C command flag "wide API support" has been at least temporarily retired in 5.8.1, and the -C has been repurposed, see *perlrun*.)

Unicode in %ENV

Currently the %ENV entries are always byte strings.

Regexps

regexp optimiser optional

The regexp optimiser is not optional. It should be configurable to be, to allow its performance to be measured, and its bugs to be easily demonstrated.

POD

POD -> HTML conversion still sucks

Which is crazy given just how simple POD purports to be, and how simple HTML can be.

Misc medium sized projects

UNITCHECK

Introduce a new special block, UNITCHECK, which is run at the end of a compilation unit (module, file, eval(STRING) block). This will correspond to the Perl 6 CHECK. Perl 5's CHECK cannot be changed or removed because the O.pm/B.pm backend framework depends on it.

optional optimizer

Make the peephole optimizer optional.

You WANT *how* many

Currently contexts are void, scalar and list. split has a special mechanism in place to pass in the number of return values wanted. It would be useful to have a general mechanism for this, backwards compatible and little speed hit. This would allow proposals such as short circuiting sort to be implemented as a module on CPAN.

lexical aliases

Allow lexical aliases (maybe via the syntax `my \ $alias = \ $foo.`

no 6

Make `no 6` and `no v6` work (opposite of `use 5.005`, etc.).

IPv6

Clean this up. Check everything in core works

entersub XS vs Perl

At the moment `pp_entersub` is huge, and has code to deal with entering both perl and XS subroutines. Subroutine implementations rarely change between perl and XS at run time, so investigate using 2 ops to enter subs (one for XS, one for perl) and swap between if a sub is redefined.

@INC source filter to Filter::Simple

The second return value from a sub in `@INC` can be a source filter. This isn't documented. It should be changed to use `Filter::Simple`, tested and documented.

bincompat functions

There are lots of functions which are retained for binary compatibility. Clean these up. Move them to `mathom.c`, and don't compile for `blead`?

Use fchown/fchmod internally

The old `perltodo` notes "This has been done in places, but needs a thorough code review. Also `fchdir` is available in some platforms."

Constant folding

The peephole optimiser should trap errors during constant folding, and give up on the folding, rather than bailing out at compile time. It is quite possible that the unfoldable constant is in unreachable code, eg something akin to `$a = 0/0 if 0;`

Tests

Make Schwern poorer

Tests for everything, At which point Schwern coughs up \$500 to TPF.

test B

A test suite for the B module would be nice.

common test code for timed bailout

Write portable self destruct code for tests to stop them burning CPU in infinite loops. Needs to avoid using `alarm`, as some of the tests are testing `alarm/sleep` or timers.

Installation

compressed man pages

Be able to install them

Make Config.pm cope with differences between build and installed perl

Relocatable perl

Make it possible to create a relocatable perl binary. Will need some collusion with `Config.pm`. We could use a syntax of ... for location of current binary?

make HTML install work

And look at the splitting of perlfunc in chunks. It needs fixing.

put patchlevel in -v

Currently perl from p4/rsync ships with a patchlevel.h file that usually defines one local patch, of the form "MAINT12345" or "RC1". The output of perl -v doesn't report that a perl isn't an official release, and this information can get lost in bugs reports. Because of this, the minor version isn't bumped up until RC time, to minimise the possibility of versions of perl escaping that believe themselves to be newer than they actually are.

It would be useful to find an elegant way to have the "this is an interim maintenance release" or "this is a release candidate" in the terse -v output, and have it so that it's easy for the pumpking to remove this just as the release tarball is rolled up. This way the version pulled out of rsync would always say "I'm a development release" and it would be safe to bump the reported minor version as soon as a release ships, which would aid perl developers.

Incremental things

Some tasks that don't need to get done in one big hit.

autovivification

Make all autovivification consistent w.r.t LVALUE/RVALUE and strict/no strict;

fix tainting bugs

Fix the bugs revealed by running the test suite with the `-t` switch (via `make test.taintwarn`).

Make tainting consistent

Tainting would be easier to use if it didn't take documented shortcuts and allow taint to "leak" everywhere within an expression.

Dual life everything

As part of the "dists" plan, anything that doesn't belong in the smallest perl distribution needs to be dual lived. Anything else can be too.

Vague things

Some more nebulous ideas

threads

Make threads more robust.

POSIX memory footprint

Ilya observed that use POSIX; eats memory like there's no tomorrow, and at various times worked to cut it down. There is probably still fat to cut out - for example POSIX passes Exporter some very memory hungry data structures.

Optimize away @_

The old perltodo notes "Look at the "reification" code in `av.c`".

switch ops

The old perltodo notes "Although we have `Switch.pm` in core, Larry points to the dormant `nswitch` and `cswitch` ops in `pp.c`; using these opcodes would be much faster."

Attach/detach debugger from running program

The old perltodo notes "With `gdb`, you can attach the debugger to a running program if you pass the process ID. It would be good to do this with the Perl debugger on a running Perl program, although I'm not sure how it would be done." `ssh` and `screen` do this with named pipes in `tmp`. Maybe we can

A decent benchmark

perlbench seems impervious to any recent changes made to the perl core. It would be useful to have a reasonable general benchmarking suite that roughly represented what current perl programs do, and measurably reported whether tweaks to the core improve, degrade or don't really affect performance, to guide people attempting to optimise the guts of perl.

readpipe(LIST)

system() accepts a LIST syntax (and a PROGRAM LIST syntax) to avoid running a shell. readpipe() (the function behind qx//) could be similarly extended.

Self ties

self ties are currently illegal because they caused too many segfaults. Maybe the causes of these could be tracked down and self-ties on all types re- instated.