NAME

threads - Perl extension allowing use of interpreter based threads from perl

SYNOPSIS

```
    use threads;

    sub start_thread {
 print "Thread started\n";
    }

    my $thread  = threads->create("start_thread","argument");
    my $thread2 = $thread->create(sub { print "I am a thread"},"argument");
    my $thread3 = async { foreach (@files) { ... } };

    $thread->join();
    $thread->detach();

    $thread = threads->self();
    $thread = threads->object( $tid );

    $thread->tid();
    threads->tid();
    threads->self->tid();

    threads->yield();

    threads->list();
```

DESCRIPTION

Perl 5.6 introduced something called interpreter threads. Interpreter threads are different from "5005threads" (the thread model of Perl 5.005) by creating a new perl interpreter per thread and not sharing any data or state between threads by default.

Prior to perl 5.8 this has only been available to people embedding perl and for emulating fork() on windows.

The threads API is loosely based on the old Thread.pm API. It is very important to note that variables are not shared between threads, all variables are per default thread local. To use shared variables one must use threads::shared.

It is also important to note that you must enable threads by doing `use threads` as early as possible in the script itself and that it is not possible to enable threading inside an `eval ""`, `do`, `require`, or `use`. In particular, if you are intending to share variables with threads::shared, you must `use threads` before you `use threads::shared` and `threads` will emit a warning if you do it the other way around.

$thread = threads->create(function, LIST)

    This will create a new thread with the entry point function and give it LIST as parameters. It will return the corresponding threads object, or `undef` if thread creation failed. The new() method is an alias for create().

$thread->join

    This will wait for the corresponding thread to join. When the thread finishes, join() will return the return values of the entry point function. If the thread has been detached, an error will be thrown.The context (void, scalar or list) of the thread creation is also the context for join(). This means that if you intend to return an array from a thread, you must use `my ($thread) = threads-new(...)>`, and that if you intend to return a scalar, you must use `my $thread = ...`.

If the program exits without all other threads having been either joined or detached, then a warning will be issued. (A program exits either because one of its threads explicitly calls exit(), or in the case of the main thread, reaches the end of the main program file.)

$thread->detach

Will make the thread unjoinable, and cause any eventual return value to be discarded.

threads->self

This will return the thread object for the current thread.

$thread->tid

This will return the id of the thread. Thread IDs are integers, with the main thread in a program being 0. Currently Perl assigns a unique tid to every thread ever created in your program, assigning the first thread to be created a tid of 1, and increasing the tid by 1 for each new thread that's created.

NB the class method `threads->tid()` is a quick way to get the current thread id if you don't have your thread object handy.

threads->object(tid)

This will return the thread object for the thread associated with the specified tid. Returns undef if there is no thread associated with the tid or no tid is specified or the specified tid is undef.

threads->yield();

This is a suggestion to the OS to let this thread yield CPU time to other threads. What actually happens is highly dependent upon the underlying thread implementation.

You may do `use threads qw(yield)` then use just a bare `yield` in your code.

threads->list();

This will return a list of all non joined, non detached threads.

async BLOCK;

`async` creates a thread to execute the block immediately following it. This block is treated as an anonymous sub, and so must have a semi-colon after the closing brace. Like `threads->new`, `async` returns a thread object.

WARNINGS

A thread exited while %d other threads were still running

A thread (not necessarily the main thread) exited while there were still other threads running. Usually it's a good idea to first collect the return values of the created threads by joining them, and only then exit from the main thread.

TODO

The current implementation of threads has been an attempt to get a correct threading system working that could be built on, and optimized, in newer versions of perl.

Currently the overhead of creating a thread is rather large, also the cost of returning values can be large. These are areas were there most likely will be work done to optimize what data that needs to be cloned.

BUGS

Parent-Child threads.

On some platforms it might not be possible to destroy "parent" threads while there are still existing child "threads".

This will possibly be fixed in later versions of perl.

tid is I32

The thread id is a 32 bit integer, it can potentially overflow. This might be fixed in a later version of perl.

Returning objects

When you return an object the entire stash that the object is blessed as well. This will lead to a large memory usage. The ideal situation would be to detect the original stash if it existed.

Creating threads inside BEGIN blocks

Creating threads inside BEGIN blocks (or during the compilation phase in general) does not work. (In Windows, trying to use fork() inside BEGIN blocks is an equally losing proposition, since it has been implemented in very much the same way as threads.)

PERL_OLD_SIGNALS are not threadsafe, will not be.

If your Perl has been built with PERL_OLD_SIGNALS (one has to explicitly add that symbol to ccflags, see `perl -V`), signal handling is not threadsafe.

AUTHOR and COPYRIGHT

Arthur Bergman <sky at nanisky.com>

threads is released under the same license as Perl.

Thanks to

Richard Soderberg <perl at crystalflame.net> Helping me out tons, trying to find reasons for races and other weird bugs!

Simon Cozens <simon at brecon.co.uk> Being there to answer zillions of annoying questions

Rocco Caputo <troc at netrus.net>

Vipul Ved Prakash <mail at vipul.net> Helping with debugging.

please join perl-ithreads@perl.org for more information

SEE ALSO

threads::shared, *perlthrtut*, *http://www.perl.com/pub/a/2002/06/11/threads.html*, *perlcall*, *perlembed*, *perlguts*