NAME

Thread::Queue - thread-safe queues

SYNOPSIS

```
    use Thread::Queue;
    my $q = new Thread::Queue;
    $q->enqueue("foo", "bar");
    my $foo = $q->dequeue;    # The "bar" is still in the queue.
    my $foo = $q->dequeue_nb; # returns "bar", or undef if the queue was
empty
    my $left = $q->pending;   # returns the number of items still in the
queue
```

DESCRIPTION

A queue, as implemented by `Thread::Queue` is a thread-safe data structure much like a list. Any number of threads can safely add elements to the end of the list, or remove elements from the head of the list. (Queues don't permit adding or removing elements from the middle of the list).

FUNCTIONS AND METHODS

new

The `new` function creates a new empty queue.

enqueue LIST

The `enqueue` method adds a list of scalars on to the end of the queue. The queue will grow as needed to accommodate the list.

dequeue

The `dequeue` method removes a scalar from the head of the queue and returns it. If the queue is currently empty, `dequeue` will block the thread until another thread `enqueues` a scalar.

dequeue_nb

The `dequeue_nb` method, like the `dequeue` method, removes a scalar from the head of the queue and returns it. Unlike `dequeue`, though, `dequeue_nb` won't block if the queue is empty, instead returning `undef`.

pending

The `pending` method returns the number of items still in the queue.

SEE ALSO

threads, *threads::shared*