

NAME

"IO::Async::Loop::Poll" – use "IO::Async" with "poll(2)"

SYNOPSIS

Normally an instance of this class would not be directly constructed by a program. It may however, be useful for running IO::Async with an existing program already using an IO::Poll object.

```
use IO::Poll;
use IO::Async::Loop::Poll;

my $poll = IO::Poll->new;
my $loop = IO::Async::Loop::Poll->new( poll => $poll );

$loop->add( ... );

while(1) {
    my $timeout = ...
    my $ret = $poll->poll( $timeout );
    $loop->post_poll;
}
```

DESCRIPTION

This subclass of IO::Async::Loop uses the poll(2) system call to perform read-ready and write-ready tests.

By default, this loop will use the underlying poll() system call directly, bypassing the usual IO::Poll object wrapper around it because of a number of bugs and design flaws in that class; namely

- <https://rt.cpan.org/Ticket/Display.html?id=93107> – IO::Poll relies on stable stringification of IO handles
- <https://rt.cpan.org/Ticket/Display.html?id=25049> – IO::Poll->poll() with no handles always returns immediately

However, to integrate with existing code that uses an IO::Poll object, a post_poll can be called immediately after the poll method that IO::Poll object. The appropriate mask bits are maintained on the IO::Poll object when notifiers are added or removed from the loop, or when they change their want_* status. The post_poll method inspects the result bits and invokes the on_read_ready or on_write_ready methods on the notifiers.

CONSTRUCTOR**new**

```
$loop = IO::Async::Loop::Poll->new( %args )
```

This function returns a new instance of a IO::Async::Loop::Poll object. It takes the following named arguments:

poll The IO::Poll object to use for notification. Optional; if a value is not given, the underlying IO::Poll::_poll() function is invoked directly, outside of the object wrapping.

METHODS**post_poll**

```
$count = $loop->post_poll
```

This method checks the returned event list from a IO::Poll::poll call, and calls any of the notification methods or callbacks that are appropriate. It returns the total number of callbacks that were invoked; that is, the total number of on_read_ready and on_write_ready callbacks for watch_io, and watch_time event callbacks.

loop_once

```
$count = $loop->loop_once( $timeout )
```

This method calls the poll method on the stored IO::Poll object, passing in the value of \$timeout,

and then runs the `post_poll` method on itself. It returns the total number of callbacks invoked by the `post_poll` method, or `undef` if the underlying `poll` method returned an error.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>