NAME

IO::Select - OO interface to the select system call

SYNOPSIS

```
use IO::Select;

$s = IO::Select->new();

$s->add(\*STDIN);
$s->add($some_handle);

@ready = $s->can_read($timeout);

@ready = IO::Select->new(@handles)->can_read(0);
```

DESCRIPTION

The `IO::Select` package implements an object approach to the system `select` function call. It allows the user to see what IO handles, see *IO::Handle*, are ready for reading, writing or have an exception pending.

CONSTRUCTOR

new ([HANDLES])

The constructor creates a new object and optionally initialises it with a set of handles.

METHODS

add (HANDLES)

Add the list of handles to the `IO::Select` object. It is these values that will be returned when an event occurs. `IO::Select` keeps these values in a cache which is indexed by the `fileno` of the handle, so if more than one handle with the same `fileno` is specified then only the last one is cached.

Each handle can be an `IO::Handle` object, an integer or an array reference where the first element is an `IO::Handle` or an integer.

remove (HANDLES)

Remove all the given handles from the object. This method also works by the `fileno` of the handles. So the exact handles that were added need not be passed, just handles that have an equivalent `fileno`

exists (HANDLE)

Returns a true value (actually the handle itself) if it is present. Returns undef otherwise.

handles

Return an array of all registered handles.

can_read ([TIMEOUT])

Return an array of handles that are ready for reading. `TIMEOUT` is the maximum amount of time to wait before returning an empty list, in seconds, possibly fractional. If `TIMEOUT` is not given and any handles are registered then the call will block.

can_write ([TIMEOUT])

Same as `can_read` except check for handles that can be written to.

has_exception ([TIMEOUT])

Same as `can_read` except check for handles that have an exception condition, for example pending out-of-band data.

count ()

Returns the number of handles that the object will check for when one of the `can_` methods is called or the object is passed to the `select` static method.

bits()

Return the bit string suitable as argument to the core select() call.

select (READ, WRITE, EXCEPTION [, TIMEOUT])

`select` is a static method, that is you call it with the package name like `new`. `READ`, `WRITE` and `EXCEPTION` are either `undef` or `IO::Select` objects. `TIMEOUT` is optional and has the same effect as for the core select call.

The result will be an array of 3 elements, each a reference to an array which will hold the handles that are ready for reading, writing and have exceptions respectively. Upon error an empty list is returned.

EXAMPLE

Here is a short example which shows how `IO::Select` could be used to write a server which communicates with several sockets while also listening for more connections on a listen socket

```
use IO::Select;
use IO::Socket;

$lsn = new IO::Socket::INET(Listen => 1, LocalPort => 8080);
$sel = new IO::Select( $lsn );

while(@ready = $sel->can_read) {
    foreach $fh (@ready) {
        if($fh == $lsn) {
            # Create a new socket
            $new = $lsn->accept;
            $sel->add($new);
        }
        else {
            # Process socket

            # Maybe we have finished with the socket
            $sel->remove($fh);
            $fh->close;
        }
    }
}
```

AUTHOR

Graham Barr. Currently maintained by the Perl Porters. Please report all bugs to <perl5-porters@perl.org>.

COPYRIGHT