NAME

"IO::Async::PID" – event callback on exit of a child process

SYNOPSIS

```
use IO::Async::PID;
use POSIX qw( WEXITSTATUS );

use IO::Async::Loop;
my $loop = IO::Async::Loop->new;

my $kid = $loop->fork(
   code => sub {
      print "Child sleeping..\n";
      sleep 10;
      print "Child exiting\n";
      return 20;
   },
);

print "Child process $kid started\n";

my $pid = IO::Async::PID->new(
   pid => $kid,

   on_exit => sub {
      my ( $self, $exitcode ) = @_;
      printf "Child process %d exited with status %d\n",
         $self->pid, WEXITSTATUS($exitcode);
   },
);

$loop->add( $pid );

$loop->run;
```

DESCRIPTION

This subclass of IO::Async::Notifier invokes its callback when a process exits.

For most use cases, a IO::Async::Process object provides more control of setting up the process, connecting filehandles to it, sending data to and receiving data from it.

EVENTS

The following events are invoked, either using subclass methods or CODE references in parameters:

on_exit `$exitcode`

Invoked when the watched process exits.

PARAMETERS

The following named parameters may be passed to `new` or `configure`:

pid => INT

The process ID to watch. Must be given before the object has been added to the containing IO::Async::Loop object.

on_exit => CODE

CODE reference for the `on_exit` event.

Once the `on_exit` continuation has been invoked, the `IO::Async::PID` object is removed from the containing IO::Async::Loop object.

METHODS

pid

```
$process_id = $pid->pid
```

Returns the underlying process ID

kill

```
$pid->kill( $signal )
```

Sends a signal to the process

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>