NAME

IPC::Run::IO −− I/O channels for IPC::Run.

SYNOPSIS

NOT IMPLEMENTED YET ON Win32! Win32 does not allow *select()* on normal file descriptors; IPC::RUN::IO needs to use IPC::Run::Win32Helper to do this.

```
use IPC::Run qw( io );

## The sense of '>' and '<' is opposite of perl's open(),
## but agrees with IPC::Run.
$io = io( "filename", '>',  \$recv );
$io = io( "filename", 'r',  \$recv );

## Append to $recv:
$io = io( "filename", '>>', \$recv );
$io = io( "filename", 'ra', \$recv );

$io = io( "filename", '<',  \$send );
$io = io( "filename", 'w',  \$send );

$io = io( "filename", '<<', \$send );
$io = io( "filename", 'wa', \$send );

## Handles / IO objects that the caller opens:
$io = io( \*HANDLE,   '<',  \$send );

$f = IO::Handle->new( ... ); # Any subclass of IO::Handle
$io = io( $f, '<', \$send );

require IPC::Run::IO;
$io = IPC::Run::IO->new( ... );

## Then run(), harness(), or start():
run $io, ...;

## You can, of course, use io() or IPC::Run::IO->new() as an
## argument to run(), harness, or start():
run io( ... );
```

DESCRIPTION

This class and module allows filehandles and filenames to be harnessed for I/O when used IPC::Run, independent of anything else IPC::Run is doing (except that errors & exceptions can affect all things that IPC::Run is doing).

SUBCLASSING

INCOMPATIBLE CHANGE: due to the awkwardness introduced in ripping pseudohashes out of Perl, this class *no longer* uses the fields pragma.

SUBROUTINES

new

I think it takes >> or << along with some other data.

TODO: Needs more thorough documentation. Patches welcome.

filename

Gets/sets the filename. Returns the value after the name change, if any.

init Does initialization required before this can be run. This includes *open()*ing the file, if necessary, and clearing the destination scalar if necessary.

open
If a filename was passed in, opens it. Determines if the handle is open via *fileno()*. Throws an exception on error.

open_pipe
If this is a redirection IO object, this opens the pipe in a platform independent manner.

close
Closes the handle. Throws an exception on failure.

fileno
Returns the fileno of the handle. Throws an exception on failure.

mode
Returns the operator in terms of 'r', 'w', and 'a'. There is a state 'ra', unlike Perl's *open()*, which indicates that data read from the handle or file will be appended to the output if the output is a scalar. This is only meaningful if the output is a scalar, it has no effect if the output is a subroutine.

The redirection operators can be a little confusing, so here's a reference table:

```
>       r       Read from handle in to process
<       w       Write from process out to handle
>>      ra      Read from handle in to process, appending it to existing
                data if the destination is a scalar.
<<      wa      Write from process out to handle, appending to existing
                data if IPC::Run::IO opened a named file.
```

op Returns the operation: '<', '>', '<<', '>>'. See "mode" if you want to spell these 'r', 'w', etc.

binmode
Sets/gets whether this pipe is in binmode or not. No effect off of Win32 OSs, of course, and on Win32, no effect after the harness is *start()*ed.

dir Returns the first character of `$self->op`. This is either "<" or ">".

poll
TODO: Needs confirmation that this is correct. Was previously undocumented.

I believe this is polling the IO for new input and then returns undef if there will never be any more input, 0 if there is none now, but there might be in the future, and TRUE if more input was gotten.

AUTHOR

Barrie Slaymaker <barries@slaysys.com>

TODO

Implement bidirectionality.