

NAME

IO::Seekable - supply seek based methods for I/O objects

SYNOPSIS

```
use IO::Seekable;
package IO::Something;
@ISA = qw(IO::Seekable);
```

DESCRIPTION

IO::Seekable does not have a constructor of its own as it is intended to be inherited by other IO::Handle based objects. It provides methods which allow seeking of the file descriptors.

`$io->getpos`

Returns an opaque value that represents the current position of the IO::File, or undef if this is not possible (eg an unseekable stream such as a terminal, pipe or socket). If the `fgetpos()` function is available in your C library it is used to implement getpos, else perl emulates getpos using C's `ftell()` function.

`$io->setpos`

Uses the value of a previous `getpos` call to return to a previously visited position. Returns "0 but true" on success, undef on failure.

See *perlfunc* for complete descriptions of each of the following supported IO::Seekable methods, which are just front ends for the corresponding built-in functions:

`$io->seek ( POS, WHENCE )`

Seek the IO::File to position POS, relative to WHENCE:

WHENCE=0 (SEEK_SET)

POS is absolute position. (Seek relative to the start of the file)

WHENCE=1 (SEEK_CUR)

POS is an offset from the current position. (Seek relative to current)

WHENCE=2 (SEEK_END)

POS is an offset from the end of the file. (Seek relative to end)

The `SEEK_*` constants can be imported from the `Fcntl` module if you don't wish to use the numbers 0 1 or 2 in your code.

Returns 1 upon success, 0 otherwise.

`$io->sysseek( POS, WHENCE )`

Similar to `$io->seek`, but sets the IO::File's position using the system call `lseek(2)` directly, so will confuse most perl IO operators except `sysread` and `syswrite` (see *perlfunc* for full details)

Returns the new position, or undef on failure. A position of zero is returned as the string "0 but true"

`$io->tell`

Returns the IO::File's current position, or -1 on error.

SEE ALSO

perlfunc, "I/O Operators" in *perlop*, *IO::Handle* *IO::File*

HISTORY

Derived from FileHandle.pm by Graham Barr <gbarr@pobox.com>