NAME

IO::String – Emulate file interface for in–core strings

SYNOPSIS

```
use IO::String;
$io = IO::String->new;
$io = IO::String->new($var);
tie *IO, 'IO::String';

# read data
<$io>;
$io->getline;
read($io, $buf, 100);

# write data
print $io "string\n";
$io->print(@data);
syswrite($io, $buf, 100);

select $io;
printf "Some text %s\n", $str;

# seek
$pos = $io->getpos;
$io->setpos(0);         # rewind
$io->seek(-30, -1);
seek($io, 0, 0);
```

DESCRIPTION

The `IO::String` module provides the `IO::File` interface for in-core strings. An `IO::String` object can be attached to a string, and makes it possible to use the normal file operations for reading or writing data, as well as for seeking to various locations of the string. This is useful when you want to use a library module that only provides an interface to file handles on data that you have in a string variable.

Note that perl–5.8 and better has built-in support for ‘‘in memory’’ files, which are set up by passing a reference instead of a filename to the *open()* call. The reason for using this module is that it makes the code backwards compatible with older versions of Perl.

The `IO::String` module provides an interface compatible with `IO::File` as distributed with *IO–1.20*, but the following methods are not available: new_from_fd, fdopen, format_write, format_page_number, format_lines_per_page, format_lines_left, format_name, format_top_name.

The following methods are specific to the `IO::String` class:

`$io` = IO::String–>new
`$io` = IO::String–>new(`$string`)

The constructor returns a newly-created `IO::String` object. It takes an optional argument, which is the string to read from or write into. If no `$string` argument is given, then an internal buffer (initially empty) is allocated.

The `IO::String` object returned is tied to itself. This means that you can use most Perl I/O built-ins on it too: readline, <>, getc, print, printf, syswrite, sysread, close.

`$io`–>open
`$io`–>open(`$string`)

Attaches an existing IO::String object to some other `$string`, or allocates a new internal buffer (if no argument is given). The position is reset to 0.

`$io->string_ref`
: Returns a reference to the string that is attached to the `IO::String` object. Most useful when you let the `IO::String` create an internal buffer to write into.

`$io->pad`
`$io->pad( $char )`
: Specifies the padding to use if the string is extended by either the *seek()* or *truncate()* methods. It is a single character and defaults to "\0".

`$io->pos`
`$io->pos( $newpos )`
: Yet another interface for reading and setting the current read/write position within the string (the normal getpos/setpos/tell/seek methods are also available). The *pos()* method always returns the old position, and if you pass it an argument it sets the new position.

  There is (deliberately) a difference between the *setpos()* and *seek()* methods in that *seek()* extends the string (with the specified padding) if you go to a location past the end, whereas *setpos()* just snaps back to the end. If *truncate()* is used to extend the string, then it works as *seek()*.

BUGS

In Perl versions < 5.6, the TIEHANDLE interface was incomplete. If you use such a Perl, then *seek()*, *tell()*, *eof()*, *fileno()*, *binmode()* will not do anything on an `IO::String` handle. See perltie for details.

SEE ALSO

IO::File, IO::Stringy, "open" in perlfunc

COPYRIGHT