NAME

File::ReadBackwards.pm −− Read a file backwards by lines.

SYNOPSIS

```
use File::ReadBackwards ;

# Object interface

$bw = File::ReadBackwards->new( 'log_file' ) or
                    die "can't read 'log_file' $!" ;

while( defined( $log_line = $bw->readline ) ) {
        print $log_line ;
}

# ... or the alternative way of reading

until ( $bw->eof ) {
        print $bw->readline ;
}

# Tied Handle Interface

tie *BW, 'File::ReadBackwards', 'log_file' or
                    die "can't read 'log_file' $!" ;

while( <BW> ) {
        print ;
}
```

DESCRIPTION

This module reads a file backwards line by line. It is simple to use, memory efficient and fast. It supports both an object and a tied handle interface.

It is intended for processing log and other similar text files which typically have their newest entries appended to them. By default files are assumed to be plain text and have a line ending appropriate to the OS. But you can set the input record separator string on a per file basis.

OBJECT INTERFACE

These are the methods in `File::ReadBackwards`' object interface:

new(`$file`, [$rec_sep], [$sep_is_regex])

`new` takes as arguments a filename, an optional record separator and an optional flag that marks the record separator as a regular expression. It either returns the object on a successful open or undef upon failure. $! is set to the error code if any.

readline

`readline` takes no arguments and it returns the previous line in the file or undef when there are no more lines in the file. If the file is a non-seekable file (e.g. a pipe), then undef is returned.

getline

`getline` is an alias for the readline method. It is here for compatibility with the IO::* classes which has a getline method.

eof

`eof` takes no arguments and it returns true when *readline()* has iterated through the whole file.

close

`close` takes no arguments and it closes the handle

tell

`tell` takes no arguments and it returns the current filehandle position. This value may be used to *seek()* back to this position using a normal file handle.

get_handle

`get_handle` takes no arguments and it returns the internal Perl filehandle used by the File::ReadBackwards object. This handle may be used to read the file forward. Its seek position will be set to the position that is returned by the *tell()* method. Note that interleaving forward and reverse reads may produce unpredictable results. The only use supported at present is to read a file backward to a certain point, then use 'handle' to extract the handle, and read forward from that point.

TIED HANDLE INTERFACE

tie(*HANDLE, 'File::ReadBackwards', `$file`, [$rec_sep], [$sep_is_regex])

The TIEHANDLE, READLINE, EOF, CLOSE and TELL methods are aliased to the new, readline, eof, close and tell methods respectively so refer to them for their arguments and API. Once you have tied a handle to File::ReadBackwards the only I/O operation permissible is <> which will read the previous line. You can call *eof()* and *close()* on the tied handle as well. All other tied handle operations will generate an unknown method error. Do not seek, write or perform any other unsupported operations on the tied handle.

LINE AND RECORD ENDINGS

Since this module needs to use low level I/O for efficiency, it can't portably seek and do block I/O without managing line ending conversions. This module supports the default record separators of normal line ending strings used by the OS. You can also set the separator on a per file basis.

The record separator is a regular expression by default, which differs from the behavior of $/.

Only if the record separator is **not** specified and it defaults to CR/LF (e.g, VMS, redmondware) will it will be converted to a single newline. Unix and MacOS files systems use only a single character for line endings and the lines are left unchanged. This means that for native text files, you should be able to process their lines backwards without any problems with line endings. If you specify a record separator, no conversions will be done and you will get the records as if you read them in binary mode.

DESIGN

It works by reading a large (8kb) block of data from the end of the file. It then splits them on the record separator and stores a list of records in the object. Each call to readline returns the top record of the list and if the list is empty it refills it by reading the previous block from the file and splitting it. When the beginning of the file is reached and there are no more lines, undef is returned. All boundary conditions are handled correctly i.e. if there is a trailing partial line (no newline) it will be the first line returned and lines larger than the read buffer size are handled properly.

NOTES

There is no support for list context in either the object or tied interfaces. If you want to slurp all of the lines into an array in backwards order (and you don't care about memory usage) just do:

```
        @back_lines = reverse <FH>.
```

This module is only intended to read one line at a time from the end of a file to the beginning.

AUTHOR

Uri Guttman, uri@stemsystems.com

COPYRIGHT