$Revision: 1.3 $ $Author: death $ $Date: 93/04/04 23:44:54 $

PSFile

INHERITS FROM File

DECLARED IN PSFile.h

CLASS DESCRIPTION

The PSFile class is a simple specilization of the File class. It merely provides several methods for interacting with a file making reference to strings and lines of text, rather than arbitrary chunks of bytes.

The current version is definitely incomplete, and only has the methods that I've direly needed so far. Most notably, it is really only usefull for *writing* to. There are no specilized reading methods.

At the moment, this allows one to append another file to the contents of the current one (usefull for importing a set of procedures and other definitions that this file will need), writing DSC comments (%%foo: ...), lines of text, and dumping a string of bytes has hex data. Additionally, it supports two methods that do printf() type of formatting. These are cruder than I'd like, and someday should be made prettier.

INSTANCE VARIABLES

Inherited from Object	Class	isa;
Declared in PSFile	None	

METHOD TYPES

From other files
- AppendFrom:

Comments
- WriteDSCComment:
- WriteFormattedDSCComment::
- WriteComment:
- WriteFormattedComment::

Data output
- WritePSLine:
- WriteFormattedPSLine:
- Write:BytesOfHexDataFrom:
- Write:InvertedBytesOfHexData:
- ForceNewLine
- WriteByteAsHex:

CLASS METHODS

None

INSTANCE METHODS

AppendFrom:

- **AppendFrom:** *file*

Inserts the entire contents of *file* into the current position of the file.

WriteDSCComment:

- **WriteDSCComment:** (CString) *comment*

Writes the specified string out as a PS DSC comment. Hence, if given `BoundingBox: 45, 67, 87,90', this will write out: `%%Boundingbox: 45, 67, 87, 90'.

WriteDSCCommentUsing:WithFormat:

- **WriteDSCCommentUsing:** (CString) *buffer* **WithFormat:** (CString) *format*, ...

This also writes a DSC comment. however, it uses a printf-like calling syntax. Arguments after the format will be used to generate the finished comment string. This form must be viewed as an intermediate form in two respects: I'm not yet sure whether I should be being passed a buffer or a maximum size of the buffer. Additionally, ideally a later version or

different method will do all this without the user passing a buffer. in any case, the buffer passed is used by the routine to build the final formatted string. If it is not long enough, the results are unpredictable. The format string is used, in printf fashion, to dictate the consumption of following arguments, and they are displayed as specified by the printf type syntax in the format string. The formatted string is then written to the file.

WriteComment:

- **WriteComment:** (CString) *comment*

Writes the specified string out as a PS comment. Hence, if given `We now start the inifinite loop', this will write out: `% We now start the infinite loop'.

WriteCommentUsing:WithFormat:

- **WriteCommentUsing:** (CString) *buffer* **WithFormat:** (CString) *format*, ...

This is identical to WriteFormatttedDSCComment:: ,save that it writes a simple PS comment, and not a DSC one.

WriteText:

- **WriteText:** (CString) *text*

This writes the specified text out to the destination file litterally.

WriteTextUsing:WithFormat:

- **WriteTextUsing:** (CString) *buffer* **WithFormat:** (CString) *format*, ...

This is like WriteFormattedComment, save that it writes out a line of text, not a comment..

WritePSLine:

- **WritePSLine:** (CString) *text*

This writes the specified text out to the destination file, terminating it with a LF.

WritePSLineUsing:WithFormat:

- **WritePSLineUsing:** (CString) *buffer* **WithFormat:** (CString) *format*, ...

This is like WriteFormattedComment, save that it writes out a line of PS code and terminates it with a newline.

ForceNewLine

- **ForceNewLine**

This little kludge merely inserts a LF character into the file.

WriteByteAsHex

- **WriteByteAsHex:** (Byte) *thebyte*

Given a byte, this will write it out as a hexadecimal number. Thus, given the byte `A', it writes out `41'.

Write:BytesOfHexDataFrom:

- **Write:** (PositiveInteger) *num* **BytesOfHexDataFrom:** (ByteString) *buffer*

This writes *num* bytes of binary data from *buffer* into the file as ascii hex data. Basically, like a repeated use of WriteByteAsHex:.

Write:InvertedBytesOfHexDataFrom:

- **Write:** (PositiveInteger) *num* **InvertedBytesOfHexDataFrom:** (ByteString) *buffer*

This writes *num* bytes of binary data from *buffer* into the file as ascii hex data. Unlike Write:BytesOfHexDataFrom:, this inverts the hex data (1's become 0's, and vice versa) before it converts it..

BUGS AND PROBLEMS

This bugger *grew*. It was not planned. And several parts of it really reflect tihs strongly. A lot could be done to enhance it. If the need arrives, I'd advocate making a 2.0 revision, say, that completely broke everything that came before. For now, however, it works even though I'm not entirely sure the Write:... methods do what I always want them to do... (thinking of endian-ness)

ENHANCEMENT IDEAS

DSC comments might take a key and a value argument ('boundingbox' and '45...', and add the colon, space or whatever is required)

CONSTANTS AND DEFINED TYPES

MODIFICATION HISTORY

$Log: PSFile.rtf,v $Revision 1.3 93/04/04 23:44:54 deathSun Apr 4 23:44:54 PDT 1993Revision 1.2 93/01/10 15:08:37 deathSun Jan 10 15:08:36 PST 1993Revision 1.1 92/07/26 13:54:15 deathInitial revision