GpHugeFile

Units

GPHugeF

Classes

EGpHugeFile
EGpHugeFileStream
TGpHugeFile
TGpHugeFileStream

Types

Global Constants

hcHFFailedToAllocateBuffer
hcHFInvalidBlockSize
hcHFInvalidHandle
hcHFInvalidSeekMode
hcHFReadInBufferedWriteMode
hcHFUnexpected
hcHFUnexpectedEOF
hcHFUnknownWindowsError
hcHFWindowsError
hcHFWriteFailed
hcHFWriteInBufferedReadMode

String Handling Routines

Various utility routines/method/classes for string handling

File/Directory Name Manipulation Routines

Various utility routines/method/classes

Components

Buttons

Labels

Legend
🟩 - Marks that the item has an associated example. (this bitmap is a hyperlink.)
🟥 - Marks that the item has documented bugs.
🟨 - Marks that the item has documented todo's.
(A todo is something which should be fixed before the next release (or "real soon"!))

Relevant to classes and interfaces only
🟩 - Marks that the class/interface has a property, method or event with examples.
🟥 - Marks that the class/interface has a property, method or event with documented bugs.
🟨 - Marks that the class/interface has a property, method or event with documented todo's.

Note that a symbol in the last group is not present if the corresponding symbol in the first group is present.

GPHugeF Unit

{button &Top,JI(`',`IDH_Unit_GPHugeF')}{button &Classes,JI(`',`IDH_UnitTopic_GPHugeF_Classes')} {button &Types,JI(`',`IDH_UnitTopic_GPHugeF_OtherTypes')} {button &Const,JI(`',`IDH_UnitTopic_GPHugeF_GlobalConstants')}
Dependencies   Legend

Interface to 64-bit file functions with some added functionality.

Description

```
   (c) 2000 Primoz Gabrijelcic
   Free for personal and commercial use. No rights reserved.

   Author           : Primoz Gabrijelcic
   Creation date    : 1998-09-15
   Last modification: 2001-07-02
   Version          : 3.07
```

Classes

EGpHugeFile
  Base exception class for all exceptions raised in TGpHugeFile and descendants.
EGpHugeFileStream
  Base exception class for exceptions created in TGpHugeFileStream.
TGpHugeFile
  Encapsulation of 64-bit file functions, supporting normal, buffered, and direct access with some additional twists.
TGpHugeFileStream
  TStream descendant, wrapping a TGpHugeFile.

Other Types

HugeInt
  Alias for int64 so it is Delphi-version-independent (as much as that is possible at all).
TGpHugeFileStreamAccess
  All possible ways to access TGpHugeFileStream.
THFError
  Result of TGpHugeFile reset and rewrite methods.
THFOpenOption
  TGpHugeFile reset/rewrite options.
THFOpenOptions
  Set of all TGpHugeFile reset/rewrite options.

Global Constants

hcHFFailedToAllocateBuffer
  Failed to allocate buffer.
hcHFInvalidBlockSize
  Invalid block size.
hcHFInvalidHandle
  Invalid file handle.
hcHFInvalidSeekMode
  Invalid 'mode' parameter passed to Seek function.
hcHFReadInBufferedWriteMode
  Read operation encountered while in buffered write mode.
hcHFUnexpected
  Exception was handled and converted to EGpHugeFile but was not expected and is not categorised.
hcHFUnexpectedEOF

Unexpected end of file.

hcHFUnknownWindowsError

Unknown Windows error.

hcHFWindowsError

Windows error.

hcHFWriteFailed

Write failed - not all data was saved.

hcHFWriteInBufferedReadMode

Write operation encountered while in buffered read mode.

Author

Primoz Gabrijelcic

EGpHugeFile Object {button &Top,JI(`',`IDH_Class_EGpHugeFile')}

Hierarchy

Base exception class for all exceptions raised in TGpHugeFile and descendants.

Unit

GPHugeF

Declaration

```
EGpHugeFile = class(Exception)
```

EGpHugeFileStream Object {button &Top,JI(`',`IDH_Class_EGpHugeFileStream')}

Hierarchy

Base exception class for exceptions created in TGpHugeFileStream.

Unit
GPHugeF

Declaration

```
EGpHugeFileStream = class(EGpHugeFile)
```

TGpHugeFile Object {button &Top,JI(`',`IDH_Class_TGpHugeFile')} {button &Properties,JI(`',`IDH_ClassTopic_TGpHugeFile_Properties')} {button &Methods,JI(`',`IDH_ClassTopic_TGpHugeFile_Methods')}

Hierarchy Properties Methods

Encapsulation of 64-bit file functions, supporting normal, buffered, and direct access with some additional twists.

Unit
GPHugeF

Declaration
`TGpHugeFile = class(TObject)`

Introduced Properties

FileDate
File date/time.

FileName
File name.

IsBuffered
True if access to file is buffered.

WindowsError
Last Windows error code.

Introduced Public Methods

BlockRead
Reads 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) from a file (or buffer if access is buffered).

BlockReadUnsafe
Reads 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) from a file (or buffer if access is buffered).

BlockWrite
Writes 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) to a file (or buffer if access is buffered).

BlockWriteUnsafe
Writes 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) to a file (or buffer if access is buffered).

Close
Closes open file.

Create
Standard TGpHugeFile constructor.

CreateEx
Extended TGpHugeFile constructor.

Destroy Override
TGpHugeFile destructor.

FileExists
Tests if a specified file exists.

FilePos
Returns file pointer position in 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods).

FileSize
Returns the size of file in 'block size' units (see 'blockSize' parameter to Reset and Rewrite methods).

Flush
Flushed file buffers.

IsOpen

Returns true if file is open.

Reset

Simplest form of Reset, emulating Delphi's Reset.

ResetBuffered

Buffered Reset.

ResetEx

Full form of Reset.

Rewrite

Simplest form of Rewrite, emulating Delphi's Rewrite.

RewriteBuffered

Buffered Rewrite.

RewriteEx

Full form of Rewrite.

Seek

Repositions file pointer.

Truncate

Truncates file at current position.

TGpHugeFile Properties

Properties Methods

In TGpHugeFile

- FileDate
- FileName
- IsBuffered
- WindowsError

TGpHugeFile Methods

Properties Methods

In TGpHugeFile

- _FilePos
- _FileSize
- _Seek
- AccessFile
- AllocBuffer
  BlockRead
  BlockReadUnsafe
  BlockWrite
  BlockWriteUnsafe
- CheckHandle
  Close
  Create
  CreateEx
  Destroy
- Fetch
  FileExists
  FilePos
  FileSize
  Flush
- FlushBuffer
- FreeBuffer
- GetDate
- InitReadBuffer
- InitWriteBuffer
  IsOpen
- LoadedToTheEOF
  Reset
  ResetBuffered
  ResetEx
  Rewrite
  RewriteBuffered
  RewriteEx
- RoundToPageSize

Seek

SetDate

Transmit
Truncate

Win32Check

FileDate property

File date/time.

Applies to

TGpHugeFile

Declaration

```
Property FileDate : TDateTime Read GetDate Write SetDate;
```

FileName property

File name.

Applies to
TGpHugeFile

Declaration
```
Property FileName : string Read hfName;
```

IsBuffered property

True if access to file is buffered.

Applies to
TGpHugeFile

Declaration
```
Property IsBuffered : boolean Read hfBuffered;
```

WindowsError property

Last Windows error code.

Applies to
TGpHugeFile

Declaration
```
Property WindowsError : DWORD Read hfWindowsError;
```

_FilePos method

Returns file pointer position in bytes.

Applies to
TGpHugeFile

Declaration
```
Function _FilePos: HugeInt;
```
Virtual

Description
Used only internally.
Returns
File pointer position in bytes.
Raises
Various - system exceptions.

Implementation
```
function TGpHugeFile._FilePos: HugeInt;
var
  off: TLargeInteger;
begin
  CheckHandle;
  off.QuadPart := 0;
  off.LowPart :=
SetFilePointer(hfHandle,off.LowPart,@off.HighPart,FILE_CURRENT);
  Win32Check(off.LowPart <> $FFFFFFFF,'_FilePos');
  Result := off.QuadPart;
End;
```

_FileSize method

Returns file size.

Applies to
TGpHugeFile

Declaration
```
Function _FileSize: HugeInt;
```
Virtual

Description
If available, returns cached size.
Returns
File size in bytes.
Raises
EGpHugeFile - on Windows errors.

Implementation
```
function TGpHugeFile._FileSize: HugeInt;
begin
  if hfCachedSize < 0 then
    hfCachedSize := FileSize;
  Result := hfCachedSize;
End;
```

_Seek method

See Also

Internal implementation of Seek method.

Applies to
TGpHugeFile

Declaration
```
Procedure _Seek(offset: HugeInt; movePointer: boolean);
```
Virtual

Description
Called from other methods, too. Moves actual file pointer only when necessary or required by caller. Handles hfoCloseOnEOF files if possible.

Parameters
offset
Offset from beginning of file in 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods).
movePointer
If true, Windows file pointer will always be moved. If false, it will only be moved when Seek destination does not lie in the buffer.

Raises
Various - system exceptions.

Implementation

```
procedure TGpHugeFile._Seek(offset: HugeInt; movePointer: boolean);
var
  off: TLargeInteger;
begin
  if (not hfBuffered) or movePointer or (not hfHalfClosed) then
    CheckHandle;
  if hfBlockSize <> 1 then
    off.QuadPart := offset*hfBlockSize
  else
    off.QuadPart := offset;
  if hfBuffered then begin
    if hfBufWrite then
      FlushBuffer
    else begin
      if not movePointer then begin
        if (off.QuadPart >= hfBufFileOffs) or
           (off.QuadPart < (hfBufFileOffs-hfBufSize)) then
          movePointer := true
        else
          hfBufOffs := {$IFNDEF D4plus}Trunc{$ENDIF}
                         (off.QuadPart-(hfBufFileOffs-hfBufSize));
      end;
      if movePointer then begin
        if hfHalfClosed then begin
          if off.QuadPart <> hfBufFileOffs then //2.26: allow seek to EOF
            CheckHandle; // bang!
        end
        else begin
          SetLastError(0);
          Win32Check(SetFilePointer(
hfHandle,off.LowPart,@off.HighPart,FILE_BEGIN)<>$FFFFFFFF,'_Seek');
        end;
        //3.02: Seek to EOF in hfHalfClosed state must not invalidate the buffer
        if not (hfHalfClosed and (off.QuadPart = hfBufFileOffs)) then begin
          hfBufFileOffs := off.QuadPart;
          hfBufFilePos  := off.QuadPart;
          hfBufOffs     := 0;
          hfBufSize     := 0;
          hfCloseOnNext := false;
        end;
      end
      else if not LoadedToTheEOF then
        hfCloseOnNext := false;
    end;
  end
  else begin
    SetLastError(0);

Win32Check(SetFilePointer(hfHandle,off.LowPart,@off.HighPart,FILE_BEGIN)<>$FFF
FFFFF,'Seek');
  end;
  hfBufFilePos := off.QuadPart;
End;
```

AccessFile method

See Also

Opens/creates a file.

Applies to
TGpHugeFile

Declaration
```
Function AccessFile(blockSize: integer; reset: boolean; diskLockTimeout:
integer; diskRetryDelay: integer; waitObject: THandle): THFError;
```
Virtual

Description
AccessFile centralizes file opening in TGpHugeFile. It will set appropriate sharing mode, open or create a file, and even retry in a case of locked file (if so required).

Parameters

blockSize
Basic unit of access (same as RecSize parameter in Delphi's Reset and Rewrite).

reset
True if file is to be reset, false if it is to be rewritten.

diskLockTimeout
Max time (in milliseconds) AccessFile will wait for lock file to become free.

diskRetryDelay
Delay (in milliseconds) between attempts to open locked file.

waitObject
Handle of 'terminate' event (semaphore, mutex). If this parameter is specified (not zero) and becomes signalled, AccessFile will stop trying to open locked file and will exit with.

Returns
Status (ok, file locked, other error).

Raises
EGpHugeFile - if 'blockSize' is less or equal to zero.

Implementation
```
function TGpHugeFile.AccessFile(blockSize: integer; reset: boolean;
  diskLockTimeout: integer; diskRetryDelay: integer;
  waitObject: THandle): THFError;
var
  start: int64;

  function Elapsed: boolean;
  var
    stop: int64;
  begin
    if diskLockTimeout = 0 then
      Result := true
    else begin
      stop := GetTickCount;
      if stop < start then
        stop := stop + $100000000;
      Result := ((stop-start) > diskLockTimeout);
    end;
  end; { Elapsed }

const
  FILE_SHARING_ERRORS: set of byte = [ERROR_SHARING_VIOLATION,
ERROR_LOCK_VIOLATION];
var
  awaited  : boolean;
  creat    : DWORD;
  shareMode: DWORD;
begin { TGpHugeFile.AccessFile }
  if blockSize <= 0 then
    raise EGpHugeFile.CreateFmtHelp(sBlockSizeMustBeGreaterThanZero,
[FileName],hcHFInvalidBlockSize);
  hfBlockSize := blockSize;
  start := GetTickCount;
  repeat
    if reset then begin
      if hfCanCreate then
        creat := OPEN_ALWAYS
      else
        creat := OPEN_EXISTING;
    end
    else
      creat := CREATE_ALWAYS;
    SetLastError(0);
    hfWindowsError := 0;
    if hfShareModeSet then begin
      if hfDesiredShareMode = $FFFF then begin
        if hfDesiredAcc = GENERIC_READ then
          shareMode := FILE_SHARE_READ
        else
          shareMode := 0
      end
      else
        shareMode := hfDesiredShareMode
    end
    else begin
      if hfDesiredAcc = GENERIC_READ then
        shareMode := FILE_SHARE_READ
      else
        shareMode := 0;
    end;
    hfHandle :=
CreateFile(PChar(hfName),hfDesiredAcc,shareMode,nil,creat,hfFlags,0);
    awaited := false;
    if hfHandle = INVALID_HANDLE_VALUE then begin
      hfWindowsError := GetLastError;
      if (hfWindowsError in FILE_SHARING_ERRORS) and (diskRetryDelay > 0) and
(not Elapsed) then
        if waitObject <> 0 then
          awaited := WaitForSingleObject(waitObject, diskRetryDelay) <>
WAIT_TIMEOUT
        else
          Sleep(diskRetryDelay);
    end
    else begin
      hfWindowsError := 0;
      hfIsOpen := true;
    end;
  until (hfWindowsError = 0) or (not (hfWindowsError in FILE_SHARING_ERRORS))
or Elapsed or awaited;
```

```
  if hfWindowsError = 0 then
    Result := hfOK
  else if hfWindowsError in FILE_SHARING_ERRORS then
    Result := hfFileLocked
  else
    Result := hfError;
  if Result = hfOK then
    AllocBuffer;
End;
```

AllocBuffer method

Allocates file buffer (after freeing old buffer if allocated).

Applies to
TGpHugeFile

Declaration

```
Procedure AllocBuffer;
```

Virtual

Description
Calculates correct buffer size for direct access files and locks buffer if required. Used only internally.

Raises
Various - system exceptions.

Implementation

```
procedure TGpHugeFile.AllocBuffer;
begin
  FreeBuffer;
  if hfBufferSize = 0 then
    hfBufferSize := BUF_SIZE;
  // round up buffer size to be the multiplier of page size
  // needed for FILE_FLAG_NO_BUFFERING access, does not hurt in other cases
  hfBufferSize := RoundToPageSize(hfBufferSize);
  SetLastError(0);
  hfBuffer :=
VirtualAlloc(nil,hfBufferSize,MEM_RESERVE+MEM_COMMIT,PAGE_READWRITE);
  Win32Check(hfBuffer<>nil,'AllocBuffer');
  if hfLockBuffer then begin
    SetLastError(0);
    Win32Check(VirtualLock(hfBuffer,hfBufferSize),'AllocBuffer');
    if hfBuffer = nil then
      raise EGpHugeFile.CreateFmtHelp(sFailedToAllocateBuffer,
[FileName],hcHFFailedToAllocateBuffer);
  end;
End;
```

BlockRead method

See Also

Reads 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) from a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure BlockRead(var buf; count: DWORD; var transferred: DWORD);
```

Parameters

buf

Buffer for read data.

count

Number of 'block size' large units to be read.

transferred

(out) Number of 'block size' large units actually read.

Raises

EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.BlockRead(var buf; count: DWORD; var transferred:
DWORD);
var
  closeNow  : boolean;
  oldBufSize: DWORD;
  trans     : DWORD;
begin
  try
    if (not hfBuffered) or (not hfHalfClosed) then
      CheckHandle;
    closeNow := hfCloseOnNext;
    if hfBlockSize <> 1 then
      count := count * hfBlockSize;
    oldBufSize := hfBufSize;
    if hfBuffered then
      Fetch(buf,count,trans)
    else begin
      SetLastError(0);
      Win32Check(ReadFile(hfHandle,buf,count,trans,nil),'BlockRead');
      hfBufFilePos := hfBufFilePos + trans;
    end;
    if hfBlockSize <> 1 then
      transferred := trans div hfBlockSize
    else
      transferred := trans;
    if hfCloseOnEOF then begin
      if closeNow then begin
        if _FilePos >= FileSize then begin
          hfLastSize := FileSize;
          CloseHandle(hfHandle);
          hfHandle := INVALID_HANDLE_VALUE;
          hfHalfClosed := true; // allow FilePos to work until TGpHugeFile.Close
          hfCloseOnNext := false;
          //3.03: reset the buffer pointer
          hfBufOffs := hfBufOffs + (oldBufSize - hfBufSize);
          //2.26: rewind the buffer for Seek to work
          hfBufSize := oldBufSize;
        end;
      end
      else
        hfCloseOnNext := (hfHandle <> INVALID_HANDLE_VALUE) and
LoadedToTheEOF;
    end;
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

BlockReadUnsafe method

See Also

Reads 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) from a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure BlockReadUnsafe(var buf; count: DWORD);
```

Parameters

buf

Buffer for read data.

count

Number of 'block size' large units to be read.

Raises

EGpHugeFile - on Windows errors or if not enough data could be read from file.

Implementation

```
procedure TGpHugeFile.BlockReadUnsafe(var buf; count: DWORD);
var
  transferred: DWORD;
begin
  BlockRead(buf,count,transferred);
  if count <> transferred then begin
    if hfBuffered then
      raise EGpHugeFile.CreateHelp(sEndOfFile,hcHFUnexpectedEOF)
    else
      Win32Check(false,'BlockReadUnsafe');
  end;
End;
```

BlockWrite method

See Also

Writes 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) to a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure BlockWrite(const buf; count: DWORD; var transferred: DWORD);
```

Parameters

buf

Data to be written.

count

Number of 'block size' large units to be written.

transferred

(out) Number of 'block size' large units actually written.

Raises

EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.BlockWrite(const buf; count: DWORD; var transferred:
DWORD);
var
  trans: DWORD;
begin
  try
    CheckHandle;
    if hfBlockSize <> 1 then
      count := count * hfBlockSize;
    if hfBuffered then
      Transmit(buf,count,trans)
    else begin
      SetLastError(0);
      Win32Check(WriteFile(hfHandle,buf,count,trans,nil),'BlockWrite');
      hfBufFilePos := hfBufFilePos + trans;
    end;
    if hfBlockSize <> 1 then
      transferred := trans div hfBlockSize
    else
      transferred := trans;
    hfCachedSize := -1;
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

BlockWriteUnsafe method

See Also

Writes 'count' number of 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods) to a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure BlockWriteUnsafe(const buf; count: DWORD);
```

Parameters

buf

Data to be written.

count

Number of 'block size' large units to be written.

Raises

EGpHugeFile - on Windows errors or if data could not be written completely.

Implementation

```
procedure TGpHugeFile.BlockWriteUnsafe(const buf; count: DWORD);
var
  transferred: DWORD;
begin
  BlockWrite(buf,count,transferred);
  if count <> transferred then begin
    if hfBuffered then
      raise EGpHugeFile.CreateFmtHelp(sWriteFailed,[FileName],hcHFWriteFailed)
    else
      Win32Check(false,'BlockWriteUnsafe');
  end;
End;
```

CheckHandle method

Checks if file is open.

Applies to
TGpHugeFile

Declaration
```
Procedure CheckHandle;
```
Virtual

Description
Called from various TGpHugeFile methods.
Raises
EGpHugeFile - if file is not open.

Implementation
```
procedure TGpHugeFile.CheckHandle;
begin
  if hfHandle = INVALID_HANDLE_VALUE then
    raise EGpHugeFile.CreateFmtHelp(sFileNotOpen,
[FileName],hcHFInvalidHandle);
End;
```

Close method

Closes open file.

Applies to
TGpHugeFile

Declaration
```
Procedure Close;
```

Description
If file is not open, do nothing.

Raises
EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.Close;
begin
  try
    if IsOpen then begin
      FreeBuffer;
      if hfHandle <> INVALID_HANDLE_VALUE then begin // may be freed in BlockRead
        CloseHandle(hfHandle);
        hfHandle := INVALID_HANDLE_VALUE;
      end;
      hfHalfClosed := false;
      hfIsOpen := false;
      hfCloseOnEOF := false;
    end;
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

Create method

Standard TGpHugeFile constructor.

Applies to
TGpHugeFile

Declaration
```
Procedure Create(fileName: string);
```

Description
Prepares file for full, share none, access.

Parameters
fileName
Name of file to be accessed.

Implementation

```
constructor TGpHugeFile.Create(fileName: string);
begin
  CreateEx(fileName,FILE_ATTRIBUTE_NORMAL,GENERIC_READ+GENERIC_WRITE,0);
  hfShareModeSet := false;
End;
```

CreateEx method

Extended TGpHugeFile constructor.

Applies to
TGpHugeFile

Declaration
```
Procedure CreateEx(fileName: string; FlagsAndAttributes: DWORD =
FILE_ATTRIBUTE_NORMAL; DesiredAccess: DWORD = GENERIC_READ+GENERIC_WRITE;
DesiredShareMode: DWORD = $FFFF);
```

Description
Caller can specify desired flags, attributes, and access mode.

Parameters
fileName
Name of file to be accessed.
FlagsAndAttributes
Flags and attributes, see CreateFile help for more details.
DesiredAccess
Desired access flags, see CreateFile help for more details.

Implementation
```
constructor TGpHugeFile.CreateEx(fileName: string; FlagsAndAttributes,
  DesiredAccess, DesiredShareMode: DWORD);
begin
  inherited Create;
  hfBlockSize        := 1;
  hfBuffer           := nil;
  hfBuffered         := false;
  hfCachedSize       := -1;
  hfDesiredAcc       := DesiredAccess;
  hfDesiredShareMode := DesiredShareMode;
  hfShareModeSet     := true;
  hfFlagNoBuf        := ((FILE_FLAG_NO_BUFFERING AND FlagsAndAttributes) <>
0);
  hfFlags            := FlagsAndAttributes;
  hfHandle           := INVALID_HANDLE_VALUE;
  hfName             := fileName;
End;
```

Destroy method

TGpHugeFile destructor.

Applies to
TGpHugeFile

Declaration
```
Procedure Destroy;
```
Override

Description
Will close file if it is still open.

Implementation

```
destructor TGpHugeFile.Destroy;
begin
  Close;
  inherited Destroy;
End;
```

Fetch method

See Also

Reads 'count' number of bytes large units from a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure Fetch(var buf; count: DWORD; var transferred: DWORD);
```

Virtual

Parameters

buf

Buffer for read data.

count

Number of bytes to be read..

transferred

(out) Number of bytes actually read..

Raises

EGpHugeFile - when trying to read while in buffered write mode.

Various - system exceptions.

Implementation

```
procedure TGpHugeFile.Fetch(var buf; count: DWORD; var transferred: DWORD);
var
  got  : DWORD;
  bufp : pointer;
  read : DWORD;
  trans: DWORD;
begin
  if hfBufWrite then
    raise EGpHugeFile.CreateFmtHelp(sReadWhileInBufferedWriteMode,
[FileName],hcHFReadInBufferedWriteMode);
  transferred := 0;
  got := hfBufSize-hfBufOffs;
  if got <= count then begin
    if got > 0 then begin // read from buffer
      Move(OffsetPtr(hfBuffer,hfBufOffs)^,buf,got);
      transferred := got;
      Dec(count,got);
      hfBufFilePos := hfBufFileOffs-hfBufSize+hfBufOffs+got;
    end;
    bufp := OffsetPtr(@buf,got);
    hfBufOffs := 0;
    if count >= hfBufferSize then begin // read directly
      read := (count div hfBufferSize)*hfBufferSize;
      if hfHalfClosed then
        trans := 0 //2.26
      else if not ReadFile(hfHandle,bufp^,read,trans,nil) then
        Exit;
      hfBufFileOffs := hfBufFileOffs+trans;
      hfBufFilePos := hfBufFileOffs;
      Inc(transferred,trans);
      Dec(count,read);
      bufp := OffsetPtr(bufp,read);
      if trans < read then
```

```
        Exit; // EOF
    end;
    // fill the buffer
    if not hfHalfClosed then begin
      if LoadedToTheEOF then
        hfBufSize := 0
      else begin
        SetLastError(0);

Win32Check(ReadFile(hfHandle,hfBuffer^,hfBufferSize,hfBufSize,nil),'Fetch');
        hfBufFileOffs := hfBufFileOffs+hfBufSize;
      end;
    end
    else begin
      //3.03: when reacing end of buffer in hfHalfClosed mode, buffer must not
      //      be invalidated
      hfBufOffs := hfBufSize;
      Exit;
    end;
  end
  else
    bufp := @buf;
  if count > 0 then begin // read from buffer
    got := hfBufSize-hfBufOffs;
    if got < count then
      count := got;
    if count > 0 then
      Move(OffsetPtr(hfBuffer,hfBufOffs)^,bufp^,count);
    Inc(hfBufOffs,count);
    Inc(transferred,count);
    hfBufFilePos := hfBufFileOffs-hfBufSize+hfBufOffs;
  end;
End;
```

FileExists method

Tests if a specified file exists.

Applies to
TGpHugeFile

Declaration
```
Function FileExists: boolean;
```

Returns
True if file exists.

Implementation

```
function TGpHugeFile.FileExists: boolean;
begin
  FileExists := SysUtils.FileExists(hfName);
End;
```

FilePos method

See Also

Returns file pointer position in 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods).

Applies to
TGpHugeFile

Declaration

```
Function FilePos: HugeInt;
```

Description
Position is retrieved from cached value.

Returns
File pointer position in 'block size' large units.

Raises
EGpHugeFile - on Windows errors.

Implementation

```
function TGpHugeFile.FilePos: HugeInt;
begin
  try
    if not hfHalfClosed then
      CheckHandle;
    if hfBlockSize <> 1 then
      Result := {$IFDEF
D4plus}Trunc{$ELSE}int{$ENDIF}(hfBufFilePos/hfBlockSize)
    else
      Result := hfBufFilePos;
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

FileSize method

See Also

Returns the size of file in 'block size' units (see 'blockSize' parameter to Reset and Rewrite methods).

Applies to
TGpHugeFile

Declaration
```
Function FileSize: HugeInt;
```

Returns
Size of file in 'block size' units.

Raises
EGpHugeFile - on Windows errors.

Implementation

```
function TGpHugeFile.FileSize: HugeInt;
var
  realSize: HugeInt;
  size    : TLargeInteger;
begin
  try
    if hfHalfClosed then
      Result := hfLastSize //2.26: hfoCloseOnEOF support
    else begin
      CheckHandle;
      SetLastError(0);
      size.LowPart := GetFileSize(hfHandle,@size.HighPart);
      Win32Check(size.LowPart<>$FFFFFFFF,'FileSize');
      if hfBufFilePos > size.QuadPart then
        realSize := hfBufFilePos
      else
        realSize := size.QuadPart;
      if hfBlockSize <> 1 then
        Result := {$IFDEF D4plus}Trunc{$ELSE}int{$ENDIF}
                    (realSize/hfBlockSize)
      else
        Result := realSize;
    end;
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

Flush method

Flushed file buffers.

Applies to
TGpHugeFile

Declaration
```
Procedure Flush;
```

Raises
EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.Flush;
begin
  CheckHandle;
  SetLastError(0);
  Win32Check(FlushBuffer,'Flush');
  SetLastError(0);
  Win32Check(FlushFileBuffers(hfHandle),'Flush');
End;
```

FlushBuffer method

Flushed file buffers (internal implementation).

Applies to
TGpHugeFile

Declaration

```
Function FlushBuffer: boolean;
```

Virtual

Returns
False if data could not be written.

Implementation

```
function TGpHugeFile.FlushBuffer: boolean;
var
  written: DWORD;
begin
  if (hfBufOffs > 0) and hfBufWrite then begin
    if hfFlagNoBuf then
      hfBufOffs := RoundToPageSize(hfBufOffs);
    Result := WriteFile(hfHandle,hfBuffer^,hfBufOffs,written,nil);
    hfBufFileOffs := hfBufFileOffs+written;
    hfBufOffs     := 0;
    hfBufFilePos  := hfBufFileOffs;
    if hfFlagNoBuf then
      FillChar(hfBuffer^,hfBufferSize,0);
  end
  else
    Result := true;
End;
```

FreeBuffer method

Frees memory buffer if allocated.

Applies to
TGpHugeFile

Declaration
```
Procedure FreeBuffer;
```
Virtual

Description
Used only internally.

Raises
Various - system exceptions.

Implementation

```
procedure TGpHugeFile.FreeBuffer;
begin
  if hfBuffer <> nil then begin
    SetLastError(0);
    Win32Check(FlushBuffer,'FreeBuffer');
    if hfLockBuffer then begin
      SetLastError(0);
      Win32Check(VirtualUnlock(hfBuffer,hfBufferSize),'FreeBuffer');
    end;
    SetLastError(0);
    Win32Check(VirtualFree(hfBuffer,0,MEM_RELEASE),'FreeBuffer');
    hfBuffer := nil;
  end;
End;
```

GetDate method

Returns file date in Delphi format.

Applies to
TGpHugeFile

Declaration
```
Function GetDate: TDateTime;
```
Virtual

Returns
Returns file date in Delphi format.
Raises
EGpHugeFile - on Windows errors.

Implementation

```
function TGpHugeFile.GetDate: TDateTime;
begin
  try
    CheckHandle;
    Result := FileDateToDateTime(FileAge(FileName));
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

InitReadBuffer method

Initializes buffer for reading.

Applies to
TGpHugeFile

Declaration
```
Procedure InitReadBuffer;
```
Virtual

Implementation

```
procedure TGpHugeFile.InitReadBuffer;
begin
  hfBufOffs     := 0;
  hfBufSize     := 0;
  hfBufFileOffs := 0;
  hfBufWrite    := false;
End;
```

InitWriteBuffer method

Initializes buffer for writing.

Applies to
TGpHugeFile

Declaration
```
Procedure InitWriteBuffer;
```
Virtual

Implementation

```
procedure TGpHugeFile.InitWriteBuffer;
begin
  hfBufSize     := 0;
  hfBufOffs     := 0;
  hfBufFileOffs := 0;
  hfBufWrite    := true;
End;
```

IsOpen method

Returns true if file is open.

Applies to
TGpHugeFile

Declaration
```
Function IsOpen: boolean;
```

Returns
True if file is open.

Implementation

```
function TGpHugeFile.IsOpen: boolean;
begin
  Result := hfIsOpen;
End;
```

LoadedToTheEOF method

Returns true if file is loaded into the buffer up to the last byte.

Applies to
TGpHugeFile

Declaration
```
Function LoadedToTheEOF: boolean;
```
Virtual

Returns
Returns true if file is loaded into the buffer up to the last byte.

Implementation

```
function TGpHugeFile.LoadedToTheEOF: boolean;
begin
  Result := (hfBufFileOffs >= (_FileSize*hfBlockSize));
End;
```

Reset method

Simplest form of Reset, emulating Delphi's Reset.

Applies to
TGpHugeFile

Declaration
```
Procedure Reset(blockSize: integer = 1);
```

Parameters
blockSize
Basic unit of access (same as RecSize parameter in Delphi's Reset and Rewrite).

Raises
EGpHugeFile - if file could not be opened.

Implementation

```
procedure TGpHugeFile.Reset(blockSize: integer);
begin
  Win32Check(ResetEx(blockSize,0,0,0,[hfoBuffered]) = hfOK,'Reset');
End;
```

ResetBuffered method

See Also

Buffered Reset.

Applies to
TGpHugeFile

Declaration
```
Procedure ResetBuffered(blockSize: integer = 1; bufferSize: integer = 0;
lockBuffer: boolean = false);
```

Description
Caller can specifiy size of buffer and require that buffer is locked in memory (Windows require that for direct access files (FILE_FLAG_NO_BUFFERING) to work correctly).

Parameters
blockSize
Basic unit of access (same as RecSize parameter in Delphi's Reset).
bufferSize
Size of buffer. 0 means default size (BUF_SIZE, currently 64 KB).
lockBuffer
If true, buffer will be locked.

Raises
EGpHugeFile - if file could not be opened.

Implementation

```
procedure TGpHugeFile.ResetBuffered(blockSize, bufferSize: integer;
  lockBuffer: boolean);
var
  options: THFOpenOptions;
begin
  options := [hfoBuffered];
  if lockBuffer then
    Include(options,hfoLockBuffer);
  Win32Check(ResetEx(blockSize,bufferSize,0,0,options) =
hfOK,'ResetBuffered');
End;
```

ResetEx method

Full form of Reset.

Applies to
TGpHugeFile

Declaration
```
Function ResetEx(blockSize: integer = 1; bufferSize: integer = 0;
diskLockTimeout: integer = 0; diskRetryDelay: integer = 0; options:
THFOpenOptions = []; waitObject: THandle = 0): THFError;
```

Description
Will retry if file is locked by another application (if diskLockTimeout and diskRetryDelay are specified). Allows caller to specify additional options. Does not raise an exception on error.

Parameters
blockSize
Basic unit of access (same as RecSize parameter in Delphi's Reset).
bufferSize
Size of buffer. 0 means default size (BUF_SIZE, currently 64 KB).
diskLockTimeout
Max time (in milliseconds) AccessFile will wait for lock file to become free.
diskRetryDelay
Delay (in milliseconds) between attempts to open locked file.
options
Set of possible open options.
waitObject
Handle of 'terminate' event (semaphore, mutex). If this parameter is specified (not zero) and becomes signalled, AccessFile will stop trying to open locked file and will exit with.

Returns
Status (ok, file locked, other error).

Implementation

```
function TGpHugeFile.ResetEx(blockSize, bufferSize: integer;
  diskLockTimeout: integer; diskRetryDelay: integer;
  options: THFOpenOptions; waitObject: THandle): THFError;
begin
  hfWindowsError := 0;
  try
    { There's a reason behind this 'if IsOpen...' behaviour. We definitely
     don't want to release file handle if ResetEx is called twice in a row as
     that could lead to all sorts of sharing problems.
     Delphi does this wrong - if you Reset file twice in a row, handle will be
     close and fill will be reopened.
   }
    if hfCloseOnEOF and IsOpen then
      Close; //2.26
    if IsOpen then begin
      if not hfReading then
        FlushBuffer;
      hfBuffered := false;
      Seek(0);
      FreeBuffer;
    end;
    hfBuffered := hfoBuffered in options;
    hfCloseOnEOF := ([hfoCloseOnEOF,hfoBuffered] * options) =
[hfoCloseOnEOF,hfoBuffered];
```

```
    hfCanCreate := hfoCanCreate in options;
    if hfBuffered then begin
      hfBufferSize := bufferSize;
      hfLockBuffer := hfoLockBuffer in options;
    end;
    if not IsOpen then
      Result :=
AccessFile(blockSize,true,diskLockTimeout,diskRetryDelay,waitObject)
    else begin
      hfBlockSize := blockSize;
      AllocBuffer;
      Result := hfOK;
    end;
    if Result <> hfOK then
      Close
    else begin
      if hfBuffered then
        InitReadBuffer;
      hfBufFilePos := 0;
      hfReading := true;
      hfHalfClosed := false;
    end;
  except
    Result := hfOK;
  end;
End;
```

Rewrite method

Simplest form of Rewrite, emulating Delphi's Rewrite.

Applies to
TGpHugeFile

Declaration
```
Procedure Rewrite(blockSize: integer = 1);
```

Parameters
blockSize
Basic unit of access (same as RecSize parameter in Delphi's Rewrite).

Raises
EGpHugeFile - if file could not be opened.

Implementation

```
procedure TGpHugeFile.Rewrite(blockSize: integer);
begin
  Win32Check(RewriteEx(blockSize,0,0,0,[hfoBuffered]) = hfOK,'Rewrite');
End;
```

RewriteBuffered method

See Also

Buffered Rewrite.

Applies to
TGpHugeFile

Declaration

```
Procedure RewriteBuffered(blockSize: integer = 1; bufferSize: integer = 0;
lockBuffer: boolean = false);
```

Description
Caller can specifiy size of buffer and require that buffer is locked in memory (Windows require that for direct access files (FILE_FLAG_NO_BUFFERING) to work correctly).

Parameters
blockSize
Basic unit of access (same as RecSize parameter in Delphi's Rewrite).
bufferSize
Size of buffer. 0 means default size (BUF_SIZE, currently 64 KB).
lockBuffer
If true, buffer will be locked.

Raises
EGpHugeFile - if file could not be opened.

Implementation

```
procedure TGpHugeFile.RewriteBuffered(blockSize, bufferSize: integer;
  lockBuffer: boolean);
var
  options: THFOpenOptions;
begin
  options := [hfoBuffered];
  if lockBuffer then
    Include(options,hfoLockBuffer);
  Win32Check(RewriteEx(blockSize,bufferSize,0,0,options) =
hfOK,'RewriteBuffered');
End;
```

RewriteEx method

Full form of Rewrite.

Applies to

TGpHugeFile

Declaration

```
Function RewriteEx(blockSize: integer = 1; bufferSize: integer = 0;
diskLockTimeout: integer = 0; diskRetryDelay: integer = 0; options:
THFOpenOptions = []; waitObject: THandle = 0): THFError;
```

Description

Will retry if file is locked by another application (if diskLockTimeout and diskRetryDelay are specified). Allows caller to specify additional options. Does not raise an exception on error.

Parameters

blockSize
Basic unit of access (same as RecSize parameter in Delphi's Rewrite).
bufferSize
Size of buffer. 0 means default size (BUF_SIZE, currently 64 KB).
diskLockTimeout
Max time (in milliseconds) AccessFile will wait for lock file to become free.
diskRetryDelay
Delay (in milliseconds) between attempts to open locked file.
options
Set of possible open options.
waitObject
Handle of 'terminate' event (semaphore, mutex). If this parameter is specified (not zero) and becomes signalled, AccessFile will stop trying to open locked file and will exit with.

Returns

Status (ok, file locked, other error).

Implementation

```
function TGpHugeFile.RewriteEx(blockSize, bufferSize: integer;
  diskLockTimeout: integer; diskRetryDelay: integer;
  options: THFOpenOptions; waitObject: THandle): THFError;
begin
  hfWindowsError := 0;
  try
    { There's a reason behind this 'if IsOpen...' behaviour. We definitely
     don't want to release file handle if ResetEx is called twice in a row as
     that could lead to all sorts of sharing problems.
     Delphi does this wrong - if you Reset file twice in a row, handle will be
     close and fill will be reopened.
   }
    if hfCloseOnEOF and IsOpen then
      Close; //2.26
    if IsOpen then begin
      hfBuffered := false;
      Seek(0);
      Truncate;
      FreeBuffer;
    end;
    hfBuffered := hfoBuffered in options;
    if hfBuffered then begin
      hfBufferSize := bufferSize;
      hfLockBuffer := hfoLockBuffer in options;
    end;
    if not IsOpen then
      Result :=
AccessFile(blockSize,false,diskLockTimeout,diskRetryDelay,waitObject)
    else begin
      hfBlockSize := blockSize;
      AllocBuffer;
      Result := hfOK;
    end;
    if Result <> hfOK then
      Close
    else begin
      if hfBuffered then
        InitWriteBuffer;
      hfBufFilePos := 0;
      hfReading := false;
      hfHalfClosed := false;
    end;
  except
    Result := hfOK;
  end;
End;
```

RoundToPageSize method

Rounds parameter next multiplier of system page size.

Applies to
TGpHugeFile

Declaration
```
Function RoundToPageSize(bufSize: DWORD): DWORD;
```
Virtual

Description
Used to determine buffer size for direct access files (FILE_FLAG_NO_BUFFERING).
Parameters
bufSize
Initial buffer size.
Returns
bufSize Required buffer size.

Implementation
```
function TGpHugeFile.RoundToPageSize(bufSize: DWORD): DWORD;
var
  sysInfo: TSystemInfo;
begin
  GetSystemInfo(sysInfo);
  Result := (((bufSize-1) div sysInfo.dwPageSize) + 1) * sysInfo.dwPageSize;
End;
```

Seek method

See Also

Repositions file pointer.

Applies to
TGpHugeFile

Declaration
```
Procedure Seek(offset: HugeInt);
```

Description
Moves actual file pointer only when necessary.

Parameters
offset
Offset from beginning of file in 'block size' large units (see 'blockSize' parameter to Reset and Rewrite methods).

Raises
EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.Seek(offset: HugeInt);
begin
  try
    _Seek(offset,false);
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

SetDate method

Sets file date.

Applies to
TGpHugeFile

Declaration
Procedure SetDate(**const** Value: TDateTime);
Virtual

Parameters
Value
new file date.

Implementation

```
procedure TGpHugeFile.SetDate(const Value: TDateTime);
var
  err: integer;
begin
  try
    CheckHandle;
    err := FileSetDate(hfHandle,DateTimeToFileDate(Value));
    if err <> 0 then
      raise EGpHugeFile.CreateFmtHelp(sFileFailed+SysErrorMessage(err),
        ['SetDate',hfName],hcHFWindowsError);
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

Transmit method

See Also

Writes 'count' number of bytes large units to a file (or buffer if access is buffered).

Applies to

TGpHugeFile

Declaration

```
Procedure Transmit(const buf; count: DWORD; var transferred: DWORD);
```

Virtual

Parameters

buf

Data to be written.

count

Number of bytes to be written.

transferred

(out) Number of bytes actually written.

Raises

EGpHugeFile - when trying to write while in buffered read mode and file pointer is not at end of file.

Various - system exceptions.

Implementation

```
procedure TGpHugeFile.Transmit(const buf; count: DWORD; var transferred:
DWORD);
var
  place  : DWORD;
  bufp   : pointer;
  send   : DWORD;
  written: DWORD;
begin
  if not hfBufWrite then begin
    //2.32: If we are at the end of file, we can switch into write mode
    if FilePos = FileSize then begin
      InitWriteBuffer;
      hfReading := false;
    end
    else
      raise EGpHugeFile.CreateFmtHelp(sWriteWhileInBufferedReadMode,
[FileName],hcHFWriteInBufferedReadMode);
  end;
  transferred := 0;
  place := hfBufferSize-hfBufOffs;
  if place <= count then begin
    Move(buf,OffsetPtr(hfBuffer,hfBufOffs)^,place); // fill the buffer
    hfBufOffs := hfBufferSize;
    hfBufFilePos := hfBufFileOffs+hfBufOffs;
    if not FlushBuffer then
      Exit;
    transferred := place;
    Dec(count,place);
    bufp := OffsetPtr(@buf,place);
    if count >= hfBufferSize then begin // transfer N*(buffer size)
      send := (count div hfBufferSize)*hfBufferSize;
      if not WriteFile(hfHandle,bufp^,send,written,nil) then
```

```
        Exit;
      hfBufFileOffs := hfBufFileOffs+written;
      hfBufFilePos := hfBufFileOffs;
      Inc(transferred,written);
      Dec(count,send);
      bufp := OffsetPtr(bufp,send);
    end;
  end
  else
    bufp := @buf;
  if count > 0 then begin // store leftovers
    Move(bufp^,OffsetPtr(hfBuffer,hfBufOffs)^,count);
    Inc(hfBufOffs,count);
    Inc(transferred,count);
    hfBufFilePos := hfBufFileOffs+hfBufOffs;
  end;
End;
```

Truncate method

Truncates file at current position.

Applies to
TGpHugeFile

Declaration
```
Procedure Truncate;
```

Raises
EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFile.Truncate;
begin
  try
    CheckHandle;
    if hfBuffered then
      _Seek(FilePos,true);
    SetLastError(0);
    Win32Check(SetEndOfFile(hfHandle),'Truncate');
  except
    on EGpHugeFile do
      raise;
    on E:Exception do
      raise EGpHugeFile.CreateHelp(E.Message,hcHFUnexpected);
  end;
End;
```

Win32Check method

Checks condition and creates appropriately formatted EGpHugeFile exception.

Applies to
TGpHugeFile

Declaration
```
Procedure Win32Check(condition: boolean; method: string);
```
Virtual

Parameters
condition
If false, Win32Check will generate an exception.
method
Name of TGpHugeFile method that called Win32Check.
Raises
EGpHugeFile - if (not condition).

Implementation
```
procedure TGpHugeFile.Win32Check(condition: boolean; method: string);
var
  Error: EGpHugeFile;
begin
  if not condition then begin
    hfWindowsError := GetLastError;
    if hfWindowsError <> ERROR_SUCCESS then
      Error := EGpHugeFile.CreateFmtHelp(sFileFailed+SWin32Error,
        [method,hfName,hfWindowsError,SysErrorMessage(hfWindowsError)],
        hcHFWindowsError)
    else
      Error := EGpHugeFile.CreateFmtHelp(sFileFailed+SUnkWin32Error,
        [method,hfName],hcHFUnknownWindowsError);
    raise Error;
  end;
End;
```

TGpHugeFileStream Object

{button &Top,JI(`',`IDH_Class_TGpHugeFileStream')} {button &Properties,JI(`',`IDH_ClassTopic_TGpHugeFileStream_Properties')} {button &Methods,JI(`',`IDH_ClassTopic_TGpHugeFileStream_Methods')}

Hierarchy Properties Methods

TStream descendant, wrapping a TGpHugeFile.

Unit
GPHugeF

Declaration
`TGpHugeFileStream = **class**(TStream)`

Description
Although it does not support huge files fully (because of TStream limitations - 'longingt' is used instead of 'int64' in critical places), you could still use it as a buffered file stream.

Introduced Properties

FileName
  Name of underlying file.

Size
  Stream size.

WindowsError
  Last Windows error code.

Introduced Public Methods

Create
  Initializes stream and opens file in required access mode.

CreateFromHandle
  Initializes stream and assigns it an already open TGpHugeFile object.

Destroy Virtual
  Destroys stream and file access object (if created in constructor).

Read Virtual
  Reads 'count' number of bytes into buffer.

Seek Virtual
  Repositions stream pointer.

Write Virtual
  Writes 'count' number of bytes to the file.

TGpHugeFileStream Properties

Properties Methods

In TGpHugeFileStream

Virtual FileName

Size

Virtual WindowsError

TGpHugeFileStream Methods

Properties Methods

In TGpHugeFileStream

Create
CreateFromHandle
Destroy

Virtual GetFileName

Virtual GetSize

Virtual GetWindowsError

Read
Seek

Virtual SetSize

Virtual Win32Check

Write

FileName property

Name of underlying file.

Applies to
TGpHugeFileStream

Declaration
```
Property FileName : string Read GetFileName;
```

Size property

Stream size.

Applies to
TGpHugeFileStream

Declaration
`Property Size : longint` **`Read`** `GetSize` **`Write`** `SetSize;`

Description
Reintroduced to override GetSize (static in TStream) with faster version.

WindowsError property

Last Windows error code.

Applies to
TGpHugeFileStream

Declaration
```
Property WindowsError : DWORD Read GetWindowsError;
```

Create method

Initializes stream and opens file in required access mode.

Applies to

TGpHugeFileStream

Declaration

```
Procedure Create(const fileName: string; access: TGpHugeFileStreamAccess;
openOptions: THFOpenOptions = [hfoBuffered]);
```

Parameters

fileName

Name of file to be accessed.

access

Required access mode.

openOptions

Set of possible open options.

Implementation

```
constructor TGpHugeFileStream.Create(const fileName: string;
  access: TGpHugeFileStreamAccess; openOptions: THFOpenOptions);
begin
  inherited Create;
  hfsExternalHF := false;
  case access of
    accRead:
      begin
        hfsFile := TGpHugeFile.CreateEx(fileName, FILE_ATTRIBUTE_NORMAL,
GENERIC_READ);
        hfsFile.Win32Check(hfsFile.ResetEx(1,0,0,0,openOptions) = hfOK,
'Reset');
      end; //accRead
    accWrite:
      begin
        hfsFile := TGpHugeFile.CreateEx(fileName, FILE_ATTRIBUTE_NORMAL,
GENERIC_WRITE);
        hfsFile.Win32Check(hfsFile.RewriteEx(1,0,0,0,openOptions) = hfOK,
'Rewrite');
      end; //accWrite
    accReadWrite:
      begin
        hfsFile := TGpHugeFile.CreateEx(fileName, FILE_ATTRIBUTE_NORMAL,
GENERIC_READ+GENERIC_WRITE);
        hfsFile.Win32Check(hfsFile.ResetEx(1,0,0,0,openOptions) = hfOK,
'Reset');
      end; // accReadWrite
    accAppend:
      begin
        hfsFile := TGpHugeFile.CreateEx(fileName, FILE_ATTRIBUTE_NORMAL,
GENERIC_READ+GENERIC_WRITE);
        hfsFile.Win32Check(hfsFile.ResetEx(1,0,0,0,openOptions) = hfOK,
'Reset');
        hfsFile.Seek(hfsFile.FileSize);
      end; //accAppend
  end; //case
End;
```

CreateFromHandle method

Initializes stream and assigns it an already open TGpHugeFile object.

Applies to
TGpHugeFileStream

Declaration

```
Procedure CreateFromHandle(hf: TGpHugeFile);
```

Parameters
hf
TGpHugeFile object to be used for data storage.

Implementation

```
constructor TGpHugeFileStream.CreateFromHandle(hf: TGpHugeFile);
begin
  inherited Create;
  hfsExternalHF := true;
  hfsFile := hf;
End;
```

Destroy method

Destroys stream and file access object (if created in constructor).

Applies to
TGpHugeFileStream

Declaration

```
Procedure Destroy;
```

Override

Implementation

```
destructor TGpHugeFileStream.Destroy;
begin
  if (not hfsExternalHF) and assigned(hfsFile) then begin
    hfsFile.Close;
    hfsFile.Free;
    hfsFile := nil;
  end;
  inherited Destroy;
End;
```

GetFileName method

Returns file name.

Applies to
TGpHugeFileStream

Declaration
```
Function GetFileName: string;
```
Virtual

Returns
Returns file name or empty string if file is not open.

Implementation

```
function TGpHugeFileStream.GetFileName: string;
begin
  if assigned(hfsFile) then
    Result := hfsFile.FileName
  else
    Result := '';
End;
```

GetSize method

Returns file size.

Applies to
TGpHugeFileStream

Declaration
```
Function GetSize: longint;
```
Virtual

Description
Better compatibility with hfCloseOnEOF files than default TStream.GetSize.
Returns
Returns file size in bytes or -1 if file is not open.

Implementation

```
function TGpHugeFileStream.GetSize: longint;
begin
  if assigned(hfsFile) then
    Result := hfsFile.FileSize
  else
    Result := -1;
End;
```

GetWindowsError method

Returns last Windows error code.

Applies to
TGpHugeFileStream

Declaration
```
Function GetWindowsError: DWORD;
```
Virtual

Returns
Last Windows error code.

Implementation

```
function TGpHugeFileStream.GetWindowsError: DWORD;
begin
  if hfsWindowsError <> 0 then
    Result := hfsWindowsError
  else if assigned(hfsFile) then
    Result := hfsFile.WindowsError
  else
    Result := 0;
End;
```

Read method

Reads 'count' number of bytes into buffer.

Applies to
TGpHugeFileStream

Declaration
```
Function Read(var buffer; count: longint): longint;
```
Override

Parameters
buffer
Buffer for read data.
count
Number of bytes to be read.
Returns
Actual number of bytes read.
Raises
EGpHugeFile - on Windows errors.

Implementation
```
function TGpHugeFileStream.Read(var buffer; count: longint): longint;
var
  bytesRead: cardinal;
begin
  hfsFile.BlockRead(Buffer,Count,bytesRead);
  Result := longint(bytesRead);
End;
```

Seek method

Repositions stream pointer.

Applies to
TGpHugeFileStream

Declaration
```
Function Seek(offset: longint; mode: word): longint;
```
Override

Parameters
offset
Offset from start, current position, or end of stream (as set by the 'mode' parameter).
mode
Specifies starting point for offset calculation (soFromBeginning, soFromCurrent, soFromEnd).
Returns
New position of stream pointer.
Raises
EGpHugeFile - on Windows errors.
EGpHugeFileStream - on invalid value of 'mode' parameter.

Implementation
```
function TGpHugeFileStream.Seek(offset: longint; mode: word): longint;
begin
  if mode = soFromBeginning then
    hfsFile.Seek(offset)
  else if mode = soFromCurrent then
    hfsFile.Seek(hfsFile.FilePos+offset)
  else if mode = soFromEnd then
    hfsFile.Seek(hfsFile.FileSize+offset)
  else
    raise EGpHugeFileStream.CreateFmtHelp(sInvalidMode,
[FileName],hcHFInvalidSeekMode);
  Result := hfsFile.FilePos;
End;
```

SetSize method

Sets stream size.

Applies to
TGpHugeFileStream

Declaration
`Procedure SetSize(newSize: longint);`
Override

Description
Truncates underlying file at specified position.
Parameters
newSize
New stream size.
Raises
EGpHugeFile - on Windows errors.

Implementation

```
procedure TGpHugeFileStream.SetSize(newSize: longint);
begin
  hfsFile.Seek(newSize);
  hfsFile.Truncate;
End;
```

Win32Check method

Checks condition and creates appropriately formatted EGpHugeFileStream exception.

Applies to
TGpHugeFileStream

Declaration
```
Procedure Win32Check(condition: boolean; method: string);
```
Virtual

Parameters
condition
If false, Win32Check will generate an exception.
method
Name of TGpHugeFileStream method that called Win32Check.
Raises
EGpHugeFileStream - if (not condition).

Implementation

```
procedure TGpHugeFileStream.Win32Check(condition: boolean; method: string);
var
  Error: EGpHugeFileStream;
begin
  if not condition then begin
    hfsWindowsError := GetLastError;
    if hfsWindowsError <> ERROR_SUCCESS then
      Error := EGpHugeFileStream.CreateFmtHelp(sStreamFailed+SWin32Error,
        [method,FileName,hfsWindowsError,SysErrorMessage(hfsWindowsError)],
        hcHFWindowsError)
    else
      Error := EGpHugeFileStream.CreateFmtHelp(sStreamFailed+SUnkWin32Error,
        [method,FileName],hcHFUnknownWindowsError);
    raise Error;
  end;
End;
```

Write method

Writes 'count' number of bytes to the file.

Applies to
TGpHugeFileStream

Declaration
```
Function Write(const buffer; count: longint): longint;
```
Override

Parameters
buffer
Data to be written.
count
Number of bytes to be written.
Returns
Actual number of bytes written.
Raises
EGpHugeFile - on Windows errors.

Implementation

```
function TGpHugeFileStream.Write(const buffer; count: longint): longint;
var
  bytesWritten: cardinal;
begin
  hfsFile.BlockWrite(buffer,count,bytesWritten);
  Result := longint(bytesWritten);
End;
```

HugeInt type

Alias for int64 so it is Delphi-version-independent (as much as that is possible at all).

Unit

GPHugeF

Declaration

```
HugeInt = LONGLONG;
```

TGpHugeFileStreamAccess type

All possible ways to access TGpHugeFileStream.

Unit
GPHugeF

Declaration
```
TGpHugeFileStreamAccess = (accRead, accWrite, accReadWrite, accAppend);
```

Values

accRead
Read access.
accWrite
Write access.
accReadWrite
Read and write access.
accAppend
Same as accReadWrite, just that Position is set immediatly after the end of file.

THFError type

Result of TGpHugeFile reset and rewrite methods.

Unit
GPHugeF

Declaration
```
THFError = (hfOK, hfFileLocked, hfError);
```

Values
hfOK
File opened successfully.
hfFileLocked
Access to file failed because it is already open and compatible sharing is not allowed.
hfError
Other file access errors (file/path not found...).

THFOpenOption type

TGpHugeFile reset/rewrite options.

Unit
GPHugeF

Declaration
```
THFOpenOption = (hfoBuffered, hfoLockBuffer, hfoCloseOnEOF, hfoCanCreate);
```

Values
hfoBuffered
Open file in buffered mode. Buffer size is either default (BUF_SIZE, currently 64 KB) or specified by the caller in ResetEx or RewriteEx methods.
hfoLockBuffer
Buffer must be locked (Windows require that for direct access files (FILE_FLAG_NO_BUFFERING) to work correctly).
hfoCloseOnEOF
Valid only when file is open for reading. If set, TGpHugeFile will close file handle as soon as last block is read from the file. This will free file for other programs while main program may still read data from TGpHugeFile's buffer. (*)
After the end of file is reached (and handle is closed):

- FilePos may be used.
- FileSize may be used.
- Seek and BlockRead may be used as long as the request can be fulfilled from the buffer.

Use of this option is not recommended when access to the file is random. (*) It was designed to use with sequential or almost sequential access to the file. hfoCloseOnEOF is ignored if hfoBuffered is not set. hfoCloseOnEOF is ignored if used in RewriteEx.
(*) hfoCloseOnEOF can cope with a program that alternately calls BlockRead and Seek requests. When BlockRead reaches EOF, this condition will be marked but file handle will not be closed yet. When BlockRead is called again, file will be closed, but only if between those calls Seek did not invalidate the buffer (Seek that can be fulfilled from the buffer is OK). This works with programs that load a small buffer and then Seek somewhere in the middle of this buffer (like Readln function in TGpTextFile class does).
hfoCanCreate
Reset is allowed to create a file if it doesn't exist.

THFOpenOptions type

Set of all TGpHugeFile reset/rewrite options.

Unit
GPHugeF

Declaration
```
THFOpenOptions = set of THFOpenOption;
```

hcHFFailedToAllocateBuffer global constant

Failed to allocate buffer.

Unit
GPHugeF

Declaration
```
hcHFFailedToAllocateBuffer = 1005;
```

hcHFInvalidBlockSize global constant

Invalid block size.

Unit
GPHugeF

Declaration
```
hcHFInvalidBlockSize = 1003;
```

hcHFInvalidHandle global constant

Invalid file handle.

Unit
GPHugeF

Declaration
```
hcHFInvalidHandle = 1004;
```

hcHFInvalidSeekMode global constant

Invalid 'mode' parameter passed to Seek function.

Unit

GPHugeF

Declaration

```
hcHFInvalidSeekMode = 1010;
```

hcHFReadInBufferedWriteMode global constant

Read operation encountered while in buffered write mode.

Unit

GPHugeF

Declaration

```
hcHFReadInBufferedWriteMode = 1007;
```

hcHFUnexpected global constant

Exception was handled and converted to EGpHugeFile but was not expected and is not categorised.

Unit

GPHugeF

Declaration

```
hcHFUnexpected = 1000;
```

hcHFUnexpectedEOF global constant

Unexpected end of file.

Unit
GPHugeF

Declaration
```
hcHFUnexpectedEOF = 1008;
```

hcHFUnknownWindowsError global constant

Unknown Windows error.

Unit

GPHugeF

Declaration

```
hcHFUnknownWindowsError = 1002;
```

hcHFWindowsError global constant

Windows error.

Unit
GPHugeF

Declaration
```
hcHFWindowsError = 1001;
```

hcHFWriteFailed global constant

Write failed - not all data was saved.

Unit
GPHugeF

Declaration
```
hcHFWriteFailed = 1009;
```

hcHFWriteInBufferedReadMode global constant

Write operation encountered while in buffered read mode.

Unit

GPHugeF

Declaration

```
hcHFWriteInBufferedReadMode = 1006;
```

Hierarchy

Exception
|
EGpHugeFile

Direct subclasses

EGpHugeFileStream

Hierarchy

Exception
|
EGpHugeFile
|
EGpHugeFileStream

Subclasses

None

Hierarchy

TObject
   |
TGpHugeFile

Subclasses

None

See Also

- Reset, Rewrite

See Also

- ResetEx, RewriteEx

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

See Also

- BUF_SIZE

See Also

- BUF_SIZE

See Also

- Reset, Rewrite

See Also

- Reset, Rewrite

Hierarchy

```
TStream
    |
TGpHugeFileStream
```

Subclasses

None