NAME

ost::MappedFile −

Create and map a disk file into memory.

SYNOPSIS

```
#include <file.h>
```

Inherits **ost::RandomFile**.

Public Member Functions

MappedFile (const char *fname, **Access** mode)
Open a file for mapping.

MappedFile (const char *fname, **Access** mode, size_t size)
Create if not exists, and map a file of specified size into memory.

MappedFile (const char *fname, **pos_t** offset, size_t size, **Access** mode)
Map a portion or all of a specified file in the specified shared memory access mode.

virtual **~MappedFile** ()
Release a mapped section of memory associated with a file.

void **sync** (void)
Synchronize the contents of the mapped portion of memory with the disk file and wait for completion.

void **sync** (char *address, size_t len)
Synchronize a segment of memory mapped from a segment fetch.

void **update** (size_t offset=0, size_t len=0)
Map a portion of the memory mapped from the file back to the file and do not wait for completion.

void **update** (char *address, size_t len)
Update a mapped region back to disk as specified by address and length.

void **release** (char *address, size_t len)
Release (unmap) a memory segment.

char * **fetch** (size_t offset=0)
Fetch a pointer to an offset within the memory mapped portion of the disk file.

char * **fetch** (off_t pos, size_t len)
Fetch and map a portion of a disk file to a logical memory block.

bool **lock** (void)
Lock the currently mapped portion of a file.

void **unlock** (void)
Unlock a locked mapped portion of a file.

size_t **pageAligned** (size_t size)
Compute map size to aligned page boundry.

Detailed Description

Create and map a disk file into memory.

This portable class works under both Posix via mmap and under the win32 API. A mapped file can be referenced directly by it’s memory segment. One can map and unmap portions of a file on demand, and update changed memory pages mapped from files immediately through **sync()**.

Author:
David Sugar <dyfet AT ostel DOT com> Map a named disk file into memory.

Constructor & Destructor Documentation

ost::MappedFile::MappedFile (const char * fname, Access mode)

Open a file for mapping. More than one segment of a file may be mapped into seperate regions of memory.

Parameters:
fname file name to access for mapping.
mode access mode to map file.

ost::MappedFile::MappedFile (const char * fname, Access mode, size_t size)
Create if not exists, and map a file of specified size into memory. **Parameters:**
fname file name to access for mapping.
mode access mode to map file.
size of file to map.

ost::MappedFile::MappedFile (const char * fname, pos_t offset, size_t size, Access mode)
Map a portion or all of a specified file in the specified shared memory access mode. Valid mapping modes include mappedRead, mappedWrite, and mappedReadWrite.

Parameters:
fname pathname of file to map into memory.
offset from start of file to begin mapping in bytes.
size of mapped area in bytes.
mode to map file.

virtual ost::MappedFile::~MappedFile () `[virtual]`
Release a mapped section of memory associated with a file. The mapped area is updated back to disk.

Member Function Documentation

char* ost::MappedFile::fetch (off_t pos, size_t len)
Fetch and map a portion of a disk file to a logical memory block. **Returns:**
pointer to memory segment.

Parameters:
pos offset of file segment to map.
len size of memory segment to map.

char* ost::MappedFile::fetch (size_t offset = `0`**)** `[inline]`
Fetch a pointer to an offset within the memory mapped portion of the disk file. This really is used for convience of matching operations between Update and Fetch, as one could simply have accessed the base pointer where the file was mapped directly.

Parameters:
offset from start of mapped memory.

bool ost::MappedFile::lock (void)
Lock the currently mapped portion of a file. **Returns:**
true if pages are locked.

size_t ost::MappedFile::pageAligned (size_t size)
Compute map size to aligned page boundry. **Parameters:**
size request.

Returns:
page aligned size.

void ost::MappedFile::release (char * address, size_t len)
Release (unmap) a memory segment. **Parameters:**
address address of memory segment to release.
len length of memory segment to release.

void ost::MappedFile::sync (char * address, size_t len)
Synchronize a segment of memory mapped from a segment fetch. **Parameters:**
address memory address to update.
len size of segment.

void ost::MappedFile::sync (void)
Synchronize the contents of the mapped portion of memory with the disk file and wait for completion. This assures the memory mapped from the file is written back.

void ost::MappedFile::unlock (void)
Unlock a locked mapped portion of a file.

void ost::MappedFile::update (char * address, size_t len)
Update a mapped region back to disk as specified by address and length. **Parameters:**
address address of segment.
len length of segment.

void ost::MappedFile::update (size_t offset = `0`, size_t len = `0`)

Map a portion of the memory mapped from the file back to the file and do not wait for completion. This is useful when mapping a database file and updating a single record.

Parameters:

offset offset into the mapped region of memory.
len length of partial region (example, record length).

Author

Generated automatically by Doxygen for GNU CommonC++ from the source code.