Memory Manager

v1.0.0

Generated by Doxygen 1.9.6

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

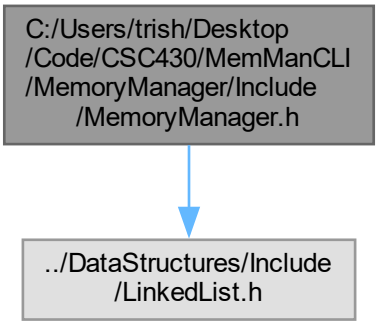
Chapter 2

File Documentation

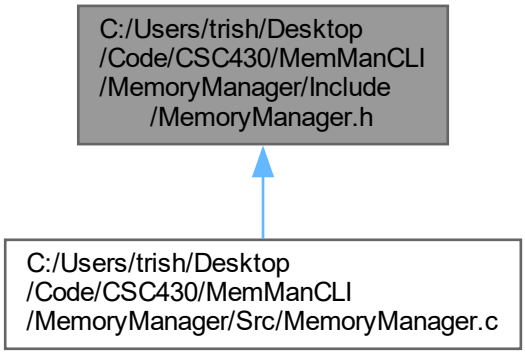
2.1 C:/Users/trish/Desktop/Code/CSC430/MemManCLI/MemoryManager/Include/MemoryManager.h File Reference

```
#include "../DataStructures/Include/LinkedList.h"
```

Include dependency graph for MemoryManager.h:



This graph shows which files directly or indirectly include this file:



Functions

- NODE ∗ createNode (size_t size, bool free)
- LIST ∗ initMemory (size_t maxSize)
- int freeMemoryLocation (LIST ∗mem, NODE ∗curr)
- void freeMemory ()
- int findFree (NODE ∗start, size_t size)
- NODE ∗ splitPage (LIST ∗mem, int pageIndex, size_t amount)
- int bestFit (LIST ∗mem, size_t amount)
- NODE ∗ requestMemory (LIST ∗mem, size_t amount)
- NODE ∗ Malloc (LIST ∗mem, size_t size)
- void DumpMemoryList (LIST ∗mem)

2.1.1 Detailed Description

Author

Tanner Ensign, Michael Vaquilar, Masaya Takahashi

Date

2/23/2023

2.1.2 Function Documentation

2.1.2.1 bestFit()

```
int bestFit (
            LIST * mem,
            size_t amount )
```

Helper method that preforms a best fit memory management algorithm in our memory manager.

Parameters

amount	amount of bytes to allocate.

Returns

a pointer to the memory location allocated.

2.1.2.2 createNode()

```
NODE * createNode (
            size_t size,
            bool Free )
```

Creates a node for the user when prompted.

Parameters

size	amount of memory.
free	is the node free of data.

Returns

pointer to node.

2.1.2.3 DumpMemoryList()

```
void DumpMemoryList (
            LIST * mem )
```

Dumps the memory to the console.

2.1.2.4 findFree()

```
int findFree (
            NODE * start,
            size_t size )
```

Finds the first free block of memory in the linked list that is the same or larger amount of size we need.

Parameters

start	node to begin the loop at.
size	of space we need.

Returns

the node with enough space.

2.1.2.5 freeMemory()

```
void freeMemory ( )
```

Frees the memory manager from the windows memory manager.

2.1.2.6 freeMemoryLocation()

```
int freeMemoryLocation (
            LIST * mem,
            NODE * curr )
```

Changes to node isFree bool to true, and then coalesces the previous and before nodes to

Parameters

start	node to begin the loop at.

2.1.2.7 initMemory()

```
LIST * initMemory (
            size_t maxSize )
```

Initalize a new memory manager object.

Parameters

maxSize	the maximum size of memory for the memory manager to use.

Returns

pointer to the memory management list.

Creates a new memory manager structure by initiliazing the busy list, free list, memory list, and queue. Sets the count equal to 0.

Returns

MEMORY structure pointer.

2.1.2.8 Malloc()

```
NODE * Malloc (
            LIST * mem,
            size_t size )
```

Allocates memory inside the memory manager created with the intialization function.

Parameters

size	of memory to allocate

Returns

pointer to the memory location of the allocation.

2.1.2.9 requestMemory()

```
NODE * requestMemory (
            LIST * mem,
            size_t amount )
```

Request memory from the memory manager linked list created when initializing the linked list. This does not use Malloc. Instead Malloc is used in initializing the memory manager, so we only use it once and split that memory up. The user can use best fit or first fit. In order to use best fit, the BEST_FIT macro at the top of the source file must be set to 1.

Parameters

amount	of memory needed.

Returns

pointer to the new node with the size requested.

Request memory from the memory manager linked list created when intializing the linked list. This does not use Malloc. Instead Malloc is used in intializing the memory manager, so we only use it once and split that memory up. The user can use best fit or first fit. In order to use best fit, the BEST_FIT macro at the top of the source file must be set to 1.

Parameters

amount	of memory needed.

Returns

pointer to the new node with the size requested.

2.1.2.10 splitPage()

```
NODE * splitPage (
            LIST * mem,
            int pageIndex,
            size_t amount )
```

Splits a page of memory if the size is larger than what is needed.

Parameters

page	to split.
amount	of memory needed.

Returns

new page;

2.2 MemoryManager.h

Go to the documentation of this file.

```
00001
00009 #ifndef MEMORYMANAGER_MEMORYMANAGER_H
00010 #define MEMORYMANAGER_MEMORYMANAGER_H
00011
00012
00013 #include "../DataStructures/Include/LinkedList.h"
00014
00015
00022 NODE *createNode(size_t size, bool free);
00023
00029 LIST *initMemory(size_t maxSize);
00030
00035 int freeMemoryLocation(LIST* mem, NODE *curr);
00036
00040 void freeMemory();
00041
00042
00043
00050 int findFree(NODE *start, size_t size);
00051
00052
00059 NODE *splitPage(LIST *mem, int pageIndex, size_t amount) ;
00060
00061
00067 int bestFit(LIST *mem, size_t amount);
00068
00078 NODE *requestMemory(LIST *mem, size_t amount);
00079
00080
00086 NODE *Malloc(LIST *mem, size_t size);
00087
00091 void DumpMemoryList(LIST *mem);
00092
00093
00094
00095
00096
00097 #endif //MEMORYMANAGER_MEMORYMANAGER_H
```

2.3 MemoryManagerTest.h

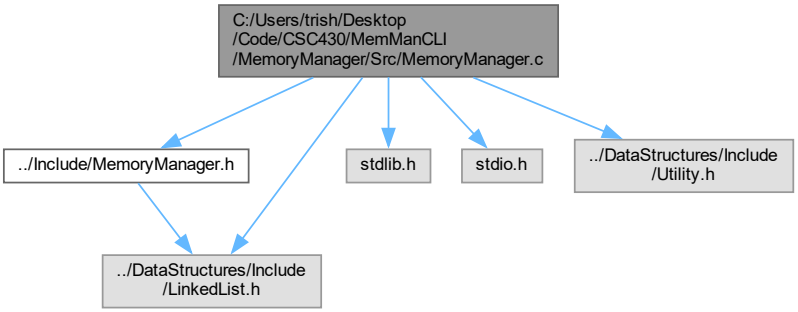
```
00001 //
00002 // Created by trish on 3/28/2023.
00003 //
00004
```

```
00005 #ifndef MEMORYMANAGER_MEMORYMANAGERTEST_H
00006 #define MEMORYMANAGER_MEMORYMANAGERTEST_H
00007
00008 void TestFirstFitCoalescing();
00009
00010 void TestBestFitCoalescing();
00011
00012 void TestAll();
00013
00014 #endif //MEMORYMANAGER_MEMORYMANAGERTEST_H
```

2.4 C:/Users/trish/Desktop/Code/CSC430/MemManCLI/MemoryManager/Src/MemoryManager.c File Reference

```
#include "../Include/MemoryManager.h"
#include <stdlib.h>
#include <stdio.h>
#include "../DataStructures/Include/Utility.h"
#include "../DataStructures/Include/LinkedList.h"
```

Include dependency graph for MemoryManager.c:



Macros

- #define **MINREQ** 0x20000
- #define **BEST_FIT** 0

Functions

- NODE ∗ createNode (size_t size, bool Free)
- LIST ∗ initMemory (size_t maxSize)
- int freeMemoryLocation (LIST ∗mem, NODE ∗curr)
- void freeMemory ()
- int findFree (NODE ∗start, size_t size)
- NODE ∗ splitPage (LIST ∗mem, int pageIndex, size_t amount)
- int bestFit (LIST ∗mem, size_t amount)
- NODE ∗ requestMemory (LIST ∗mem, size_t amount)
- NODE ∗ Malloc (LIST ∗mem, size_t size)
- void DumpMemoryList (LIST ∗mem)

Variables

- LIST * manager

2.4.1 Detailed Description

Author

Tanner Ensign, Michael Vaquilar, Masaya Takahashi

Date

2/23/2023

2.4.2 Function Documentation

2.4.2.1 bestFit()

```
int bestFit (
            LIST * mem,
            size_t amount )
```

Helper method that preforms a best fit memory management algorithm in our memory manager.

Parameters

amount	amount of bytes to allocate.

Returns

a pointer to the memory location allocated.

2.4.2.2 createNode()

```
NODE * createNode (
            size_t size,
            bool Free )
```

Creates a node for the user when prompted.

Parameters

size	amount of memory.
free	is the node free of data.

Returns

pointer to node.

2.4.2.3 DumpMemoryList()

```
void DumpMemoryList (
            LIST * mem )
```

Dumps the memory to the console.

2.4.2.4 findFree()

```
int findFree (
            NODE * start,
            size_t size )
```

Finds the first free block of memory in the linked list that is the same or larger amount of size we need.

Parameters

start	node to begin the loop at.
size	of space we need.

Returns

the node with enough space.

2.4.2.5 freeMemory()

```
void freeMemory ( )
```

Frees the memory manager from the windows memory manager.

2.4.2.6 freeMemoryLocation()

```
int freeMemoryLocation (
            LIST * mem,
            NODE * curr )
```

Changes to node isFree bool to true, and then coalesces the previous and before nodes to

Parameters

start	node to begin the loop at.

2.4.2.7 initMemory()

```
LIST * initMemory (
            size_t maxSize )
```

Creates a new memory manager structure by initiliazing the busy list, free list, memory list, and queue. Sets the count equal to 0.

Returns

MEMORY structure pointer.

2.4.2.8 Malloc()

```
NODE * Malloc (
            LIST * mem,
            size_t size )
```

Allocates memory inside the memory manager created with the intialization function.

Parameters

size	of memory to allocate

Returns

pointer to the memory location of the allocation.

2.4.2.9 requestMemory()

```
NODE * requestMemory (
            LIST * mem,
            size_t amount )
```

Request memory from the memory manager linked list created when intializing the linked list. This does not use Malloc. Instead Malloc is used in intializing the memory manager, so we only use it once and split that memory up. The user can use best fit or first fit. In order to use best fit, the BEST_FIT macro at the top of the source file must be set to 1.

Parameters

amount	of memory needed.

Returns

pointer to the new node with the size requested.

2.4.2.10 splitPage()

```
NODE * splitPage (
            LIST * mem,
            int pageIndex,
            size_t amount )
```

Splits a page of memory if the size is larger than what is needed.

Parameters

page	to split.
amount	of memory needed.

Returns

new page;

2.4.3 Variable Documentation

2.4.3.1 manager

```
LIST* manager
```

The memory manager created.

Index