```
.\"Modified from man(1) of FreeBSD, the NetBSD mdoc.template, and
mdoc.samples.
.\"See Also:
.\"man mdoc.samples for a complete listing of options
.\"man mdoc for the short list of editing options
.\"/usr/share/misc/mdoc.template
.Dd 8/4/10               \" DATE
.Dt Project 2      \" Program name and manual section number
.Sh Memory Management and Layering                \" Section Header -
required - don't modify
.Sh SYNOPSIS             \" Section Header - required - don't modify
The goal of this project is to: create a memory manager that can
initialize, track, allocate and deallocate memory.
.Sh DESCRIPTION          \" Section Header - required - don't modify
The steps taken/ functions created are as follows:
.Sh FILES                \" File used or created by the topic of the man
page
.Bl -tag -width "/Users/joeuser/Library/really_long_file_name" -compact
.It Pa /home/reptilian/MemoryManager


Directory to hold MemoryManger header and cpp file as instructed.


.It Pa /home/reptilian/MemoryManger/MemoryManager.h


Header holds prototypes, variables, and custom object struct for linked
list implementation, featuring block details and linked list for hole
merging using pointer manipulation.


.It Pa /home/reptilian/MemoryManager/MemoryManager.cpp


CPP file houses function definitions, excluding getter/setters, detailed
here:


.It Pa void MemoryManager::initialize(size_t sizeInWords)


Functionality: Initializes block with specified size (not exceeding 65536)
and linked list, triggers shutdown() to clear prior data if already
active.


How: Generates array of total size(wordsize*sizeInWords), sets linkedList
head pointer as hole of size(totalSize).



.It Pa void *MemoryManager::allocate(size_t sizeInBytes)
```

Functionality: Allocates/adjusts memor, returns nullptr if uninitialized.

How: Determines allocation block index using getList/allocator function, scans linked list for matching hole. Once found, allocates block, splits hole by creating a new memoryBlock object and adjusting necessary pointers.

Note: Calls getList(), requires separate deallocation as getList() can't handle deallocation itself.

.It Pa void MemoryManager::free(void *address)

Functionality: Frees requested memory block/ manages hole merging.

How: Scans linked list to locate allocated block by starting address and deallocates. Manages hole merging with adjacent blocks if needed. Adjusts sizes and pointers to merge holes. If previous block is hole, it adjusts size/pointers. If next block is hole: adjusts current block's size/pointers.

Note: Lacking a prev pointer in object, manual saving of the prevBlock pointer is necessary. Newly created memory must be deleted.

.It Pa int MemoryManager::dumpMemoryMap(char *filename)

Functionality: Utilizes POSIX to print hole information in specified format, returning 0=success, -1=failure.

How: Employs POSIX to print to a given filename. Utilizes getList() to fetch holes and iterates, printing required information using dprintf.

Note: Retrieves size from getList() at 0 and iterates through every other element (each hole spans 2 elements in getList array)

.It Pa void *MemoryManager::getList()

Functionality: Generates 2-byte array of hole information as [size, address_1, size_1, etc.].

How: Iterates through memory list, identifying holes (currBlock) and stores address/length in vector, counting number of holes. Creates 2-byte array (size=vector_size + 1) by first including number of holes, then all information from populated vector. Returns array.

Note: Return type must be uint16_t (2-bytes)

.It Pa void *MemoryManager::getBitmap()

Functionality: Provides bitstream array of memoryBlock information. 1 represents allocated block, 0 represents hole. The first 2 bits denote size in little-endian format, subsequent bits mirror block information.

How: Calculates total byte count based on totalBytes (rounding up to fill last byte). Generates bitstream array of uint8_t (size=totalBytes + 2 for size). Stores total byte count in little-endian format in first 2 elements. Iterates through memory list, using bitwise operations to assign 1/0 based on isAllocated. Fills in 0 for any remaining bits in last byte, mirrors each byte individually (excluding first 2 bytes).

Note: Count tracks the bit position (0-7), and index monitors the byte position.

.It Pa int bestFit(int sizeInWords, void *list)
.It Pa int worstFit(int sizeInWords, void *list)

Functionality: Provides the word offset of hole selected using the best/worst fit. Returns -1 if there's no suitable fit.

How: Receives hole information from getList, identifies hole fitting sizeInWords following the best/worst fit approach.

- Bestfit updates index based on currentBestSize (init = INT_MAX)

- Worstfit updates index based on currentWorstSize (init = -1).

Note: Not a part of the MemoryManager class.

.Sh TESTING

Given Testfile: CommandLineTest.cpp

Tests all functions

Create a library (MemoryManagement directory) and a makefile containing the build commands (g++ -g -o MemoryManager -c MemoryManager.cpp) create build.sh file to run all tests simultaneuously with Valgrind

Contents of build.sh:

```
cd MemoryManager
make
ar cr libMemoryManager.a MemoryManager
cd ..
c++ -std=c++17 -g -o ./CommandLineTest CommandLineTest.cpp -L
./MemoryManager -lMemoryManager
valgrind --leak-check=full -s ./CommandLineTest
```

Peer Testfile:

Tests a bunch of edge cases like:
- freeing ends of the memoryBlock
- switching allocators
- allac required words

Valgrind:

Ran with above tests, checks for memory leaks/errors

.Sh LINK

Unlisted Link:

https://youtu.be/7jhRSxpkiR0

.Sh REFERENCES/CITATIONS
POSIX (dumpMemoryMap):

https://notes.shichao.io/apue/ch3/

https://www.classes.cs.uchicago.edu/archive/2017/winter/51081-
1/LabFAQ/lab2/fileio.html

General aid:

https://chat.openai.com/
.Sh BUGS \" Document known, unremedied bugs

N/a
.Sh AUTHOR

Ivan Saldarriaga