CS416 - Assignment 2

Names: Nicholas Prezioso, Benjamin Cahnbley, and Marcella Alvarez
Username of iLab: njp107, bc499, and ma1143
iLab Server: protoype

New Methods:

Checks the swap file for any pages with a tid matching the current thread and swaps them in
static void swapMemoryPages();

Check a page either in memory or the swap file to see how many pages its first block takes up
static unsigned int getNumPages(unsigned int pageNum);
static unsigned int getNumPagesSwap(unsigned int pageNum);

Protects and unprotects pages based on their tid when compared to the tid of the currently running thread
static void updateMemoryProtections();

Initial setup for memory and the swap file
static void setupMemory();

Internal function which uses bitwise operations to assemble 4 characters into an integer
static unsigned int fourCharToInt(char a, char b, char c, char d);

Returns the tid of a page in either memory or the swap file
static my_pthread_t getPageTid(unsigned int pageNum);
static my_pthread_t getPageTidSwap(unsigned int pageNum);

Sets the tid of a page in either memory or the swap file
static void setPageTid(unsigned int pageNum, my_pthread_t tid);
static void setPageTidSwap(unsigned int pageNum, my_pthread_t tid);

Get or set the corresponding real location of a page in the swap file
static my_pthread_t getPageLocationSwap(unsigned int pageNum);
static void setPageLocationSwap(unsigned int pageNum, unsigned int loc);

Gets the size of the block whos metadata is poined to by i in either memory or the swap file
static unsigned int getBlockSize(int i);
static unsigned int getBlockSizeSwap(int i);

Sets the size of the block whos metadata is poined to by i in either memory or the swap file
static void setBlockSize(int i, unsigned int capacity);
static void setBlockSizeSwap(int i, unsigned int capacity);

Checks if the page or block whos metadata is pointed to by i is allocated in either memory or the swap file
static char isAllocated(int i);
static char isAllocatedSwap(int i);

Checks if a page in either memory or the swap file has enough room for an allocation
static char hasSpace(int pageName, unsigned int capacity);
static char hasSpaceSwap(int pageName, unsigned int capacity);

Rounds a double up
static int roundUp(double num);

Our implementation of malloc
void* myallocate(int capacity, char* file, int line, char threadreq);

Our implementation of free
void mydeallocate(void* toBeFreed, char* file, int line, char threadreq);

Our implementation of shared memory allocations
void* shalloc(size_t size);

Implementation:
We have implemented phases A through D to the best of our knowledge. The description of the assignment was a bit vague, so note that we made the assumption that no allocations will be larger than the size of a page minus 10 bytes (for our metadata). All requests to allocate amounts larger than this will return NULL.

Structure:
The following is the structure of metadata within memory and the swap file:
Page Layout:
1 byte for allocated/unallocated
4 bytes for TID
Beginning of blocks
Block Layout:
Metadata:
1 byte for allocated/unallocated
4 bytes for block size
Data

Swap File Page Layout:

1 byte for allocated/unallocated
4 bytes for TID
4 bytes for location in real memory
Beginning of blocks
Block Layout:
Metadata:
1 byte for allocated/unallocated
4 bytes for block size
Data

Testing:

We have tested our library extensively using numerous testers that we wrote in addition to the benchmarks provided for part 1. All tests that we have tried ran without errors.