Matthew Kelleher

November 7, 2022

CMPSC 473

<u>Project 2 Report</u>

Design of scheduler / Implementation choices and restrictions:

Date Structures:

<u>LIST</u>: *MEMORY_CHUNK** head, *MEMORY_CHUNK** tail, *int* size

<u>MEMORY_CHUNK</u>: *int* size, *int* start, *int* end, *MEMORY_CHUNK** next,

<u>If memory chunk is slab</u>: *int* type, *int* num_onj, *int* num_used, *LIST** free_mem, *void** slab_ptr

<u>MEMORY</u>: *malloc_type* type, *int* size, *void** mem_ptr, *void** start_ptr, *LIST** free_mem, *LIST** slab_table

my_setup:

Calls the new_mem function to set up a new memory data structure with the passed in parameters (type, mem_size, *start_of_memory). This method sets all passed in parameter to there respective variables in the memory data structure. Next, the list of free memory is allocated, containing the free memory lists of each power of two size under and including the memory size, is populated with empty lists, and a free memory chunk of the memory's size is set to free within the list. Finally, if the memory allocation policy is slab allocation the slab, a slab table is allocated.

my_malloc (Buddy):

If the allocation policy is buddy, the buddy_allocate method is called with the size of the memory to be allocated as a parameter to get a return pointer to the start of the memory allocated. Within the method first the header size is added to the total size and the size is rounded up to the next power of two. Next, it checks if there is a free memory chunk of the needed size within memories free_mem list using the power of two as the index of the list. If there is already a memory chunk to allocate of the needed size we continue, if not we divide up memory chunks from higher powers of two till we have a memory chunk of the needed size. If we are unable to get a memory chunk of the needed size NULL is returned. From here the found memory chunk is removed from free memory, the first 8 bits are set as the size of the memory chunk allocated, and a pointer to the start of the memory after the header is returned.

my_free (Buddy):

If the allocation policy is buddy, the buddy_free method is called with a pointer to the start of the memory to be freed as a parameter. Using the pointer, we find the start point of the memory chunk and its size which is stored in the header. Using these we add the memory chunk back into the free memory list of its given size. Finally, we check if the memory chunks buddy is also free, if it is we merge them till there is no more buddies availed to merge.

my_malloc (Slab):

If the allocation policy is slab, the slab_allocate method is called with the size of the memory to be allocated as a parameter to get a return pointer to the start of the memory allocated. Within the method first the header size is added to the total size and then we look for a slab within memories slab table that can fit a memory chunk of the given size. If one doesn't exist, we call buddy allocate to allocate a slab that can fit 64 memory chunks of the given size and add it to the slab table. If it can not be allocated, then NULL is returned. From here the first free memory chunk is removed from the free memory of the slab, the first 8 bits are set as the size of the memory chunk allocated, and a pointer to the start of the memory after the header is returned.

my_free (Slab):

If the allocation policy is slab, the slab_free method is called with a pointer to the start of the memory to be freed as a parameter. Using the pointer, we find the start point of the memory chunk and its size which is stored in the header. Using these we find the slab it is in and add the memory chunk back into the free memory list of the slab. Finally, we check if the slab is now empty, if it is we remove it from the slab table and do a buddy_free on pointer to the slab.

Issues not implemented:

All 3 functions (my_setup, my_malloc, and my_free) have been implemented fully passing all given test cases and all personal test cases.

Distribution of the workload within the team:

The project was done entirely by myself, all functions were created and implemented on my own.