Suva Shahria
Ss2479

PATRICK S. CURRIE
netid: psc62

We used ILab Specification

We made a separate library.

Our 8MB of physical memory is split into 2048 pages. At the very beginning of every page we store it's page struct. Then immediately after we store it's memory structs. The memory struct 's hold information such as memory address and size of memory written in that struct. After allocating memory below a memory struct inside the page we check to see if there is room for another memory struct, if there is we make it. If a page doesn't have enough memory we try to find another page that's assigned to the thread to fill up. Each memory struct inside the page has a head memory struct at the top of a linked list for the memory structs.
We also have a enum to differentiate between pages that are called from our library. If our library calls a malloc we assign it a page with the library enum. Now no threads have access to that page. Each time library calls malloc it tries to write to that page. If it's full we allocate a new page.

Part B.
When there is a malloc of a size greater than the size of a page enough pages are assigned to the thread to meet the demand. The pages are moved to the front of the physical memory which allows for continuous allocation of memory.
When a thread tries to access a page it doesn't have access to our signal handler moves all it's pages to the front of memory
In order to do this our thread and pages carry information about which frame they are on in the physical memory and a pointer to the head of the linked list.

Part C.
We have a counter to know when we run out of virtual memory. This counter also indexes the next location to write in the virtual memory. When we want to swap a page in we swap the head of the page to the front of physical memory. Then we go through the linked list provided by next_owner_page in the page struct and put them one after each other after the head.. This linked list holds the pages in the order they were allocated.
To write in a page we use the counter to see the next free space in the txt file. We then write it in that location.
We have commented out the swap_pages function.

Part D.
We have a pointer to the beginning 4 pages and remove access to them in parts A-B. A call to shalloc updates the pointer as it moves forward and returns the memory address previously.