# CS3502 Project 01-Part III Spring 2023

# OS Project - Memory Manager

# Learning Objectives:

## The purpose of this assignment is to continue the development examples of setting up an Operating System kernel.

## Submissions will be placed here and they will contribute to the total assignment grade of 100.

## This is the kernel build process and constitutes the first module the Memory Management Routine, before a process can be created and scheduled the kernel must allocate (and own) a block of memory from which it can distribute memory to other processes:

**MEMORY MANAGER**

**50 points**

## Starting with 08\_shell from previous assignment:

## **Create a data-structure Linked-List to store memory allocations where each node is the structure:**

## node {

## unsigned int base\_address; // Start of Memory

## unsigned int limit\_offset; // Number of bytes available

## node\* next; // Next Node

## node\* previous; // Previous Node

## }

## As you can see this will be a double linked list. Also note values with a preceding “0x” means those values are hexadecimal.

## Example:

## If base address = 0x10000 and limit\_offset = 0x0400 then the useable memory is 0x10000 – 0x103FF (which occupies 1024 bytes).

## Linked list should be searchable for (id).

## Write a function to search list by a given id:

## **e.g. node find( node \*list, int search\_address);**

## Linked list should be sortable by (base\_address) or by (limit\_offset).

## **e.g. void sort\_by\_id( node \*list);**

## **e.g. void sort\_by\_size( node \*list, bool smallest\_to\_largest);**

## Linked list should be able to add a new member in correct sorted position.

## **e.g. void add( node \*head, unsigned int base\_address, unsigned int limit\_offset);**

## **e.g. void add( node \*head, node \*new\_node);**

## Linked list should be able to delete an existing member by id.

## **e.g. void delete\_by\_address( node \*head, int delete\_id);**

## Linked list should be able to modify an existing member and resort afterwards.

## **e.g. void resize( node \*head, usigned int base\_address, unsigned int limit\_offset);**

## These function example definitions are just a template to give you an idea, if you want to use a Class or some other structure underneath to hide the maintenance parameters instead [such as \*head, \*last, \*current, memory minimum, memory maximum etc.] that is acceptable so long as it meets the functional requirements.

## The link listed is intended to create a sorted list of allocations to memory and the structure above (node) will be modified to take on new data values and eventually become a structure for a Process Control Block with a separate structure for memory allocation.

## **Once you have a linked list working you will need a second linked list to identify what are known as holes. For a given range of memory e.g. [0x10000, 0x40000] write a function that finds the unallocated spaces in memory. For example if you have the following linked list, write a function to list both holes and allocations as an output:**

## NULL <-head( 0x10000, 0x400) ⬄ node( 0x11000, 0x1000) ⬄ node( 0x20000, 0x400) -> NULL

## **void create\_holes\_list( node \*holes\_list, node\* memory\_list);**

## Then the holes\_list would be a linked list of:

## NULL <-head( 10400, 0xC00) ⬄ node( 0x12000, 0xE000) ⬄ node( 0x20400, 0x19C00) -> NULL

## This would produce a memory map of:

## **0x00000- 0x0FFFF Unavailable (i.e. used by kernel) 64kb**

## **0x10000- 0x103FF Memory allocation id 0 of 1kb**

## **0x10400- 0x10FFF Unused (Hole) of size of 3kb**

## **0x11000- 0x11FFF Memory allocation id 2 of 4kb**

## **0x12000- 0x1FFFF Unused (Hole) of size 56kb**

## **0x20000- 0x203FF Memory allocation id 10 of 1 kb**

## **0x20400- 0x3FFFF Unused (Hole) of size 127kb**

## **Total Physical Memory: 192kb**

## **Total Free: 186kb**

## **Total Allocated: 6kb**

## **Number of allocations: 3**

## **Number of free gaps: 3**

## You will then use Holes list to support future calls for memory allocation by finding space to put processes into.

## **Each time a change is made (add/delete/resize) update the memory\_list and holes\_list. Write functions to update and resort when an add/delete or resize is made to the memory list.**

## **(4) Modify the original kmalloc(u32 size, int align, u32 \*phys\_addr) found in lib\_c to allocate memory using the above linked lists to find the next available allocation space using the linked lists above instead of keeping track of next\_free. Implement kmalloc to support the three allocations fit types.**

## kmalloc according to FIRST\_FIT = 0.

## kmalloc according to BEST\_FIT = 1

## kmalloc according to WORST\_FIT = 2

## kmalloc will pick the address and return the address to the caller or return 0 if it fails:

## **u32 \*kmalloc(u32 memory\_size, int align, u32 align\_size, int fit\_type);**

## Where align == 0 means don’t align memory allocations along

1. **Write an accounting program to provide the following information for a given linked list of allocations and holes:**

print\_memory( node \*memory\_list, node \*hole\_list);

## **Total Physical Memory: 192kb**

## **Total Free: 186kb**

## **Total Allocated: 6kb**

## **Number of allocations: 3**

## **Number of free gaps: 3**

## **Start of Memory: 0x10000**

## **End of Memory: 0x3FFFF**

## Optional +10 points - Write a function to show the Memory Map as above with holes and allocated:

## **0x00000- 0x0FFFF Unavailable (i.e. used by kernel) 64kb**

## **0x10000- 0x103FF Memory allocation id 0 of 1kb**

## **0x10400- 0x10FFF Unused (Hole) of size of 3kb**

## **0x11000- 0x11FFF Memory allocation id 2 of 4kb**

## **0x12000- 0x1FFFF Unused (Hole) of size 56kb**

## **0x20000- 0x203FF Memory allocation id 10 of 1 kb**

## **0x20400- 0x3FFFF Unused (Hole) of size 127kb**

## **SUBMISSION DATA:**

## **Modify kernel.c to allocate at least 5 blocks of memory of varying sizes with a test pattern of:**

## **Have the program output your name and class section**

## **List the current allocation (none)**

## **Allocate at least 5 memory blocks**

## **List the current allocation**

## **Delete a head block**

## **List the current allocation**

## **Delete the tail block**

## **List the current allocation**

## **Add two blocks**

## **Delete an address that hasn’t been allocated**

## **List the current allocation**

## **Delete a block in the middle**

## **List the current allocation**

## **Delete all blocks**

## **Delete an address that hasn’t been allocated**

## **List the current allocation**

## **Have the program output your name and class section**

## **Show screenshots or provide a text log output of the results.**

## **If you did the optional memory map dump a memory map and show it in a text file or screenshout.**

## **Provide the entire 09\_memory directory as a tar and gz file in D2L.**

## **For the interested, see the guide website I used as a template for this: https://github.com/cfenollosa/os-tutorial**

**Submission Guidelines:**

* No handwritten submission is accepted, always submit within this document file. Exception, If you are submitting code, submit as an ASCII formatted text file such as .java, .py, .cpp, .cs etc. or even .txt. NO ZIP FILES.
* You may include your freehand drawing/image and handwritten scans in the submission. However, the writing and images must be clearly legible.  If I notice them as illegible to me prior to the due date I will let you know to provide clearer version.  This will not change the due date.  It is best to present non-handwritten submissions, generally, as is done in the professional setting.
* Show all work/calculations.
* **Please complete your entire work in a single Word Document and Save the file as: yournetid\_CS3502\_Project0X.docx; otherwise, 25% points will be deducted. For instance, I would save my file as: ogarcia5\_CS3502\_Project0X.docx. Then upload your file in D2L.**
* Please observe the submission due date and time. After the due date there is a 50% penalty for the next 24 hours. Any submission after 24 hours of the due date will be graded at 0%.
* If you include a reference or an image taken from other sources, please cite them appropriately. APA is preferred, but cite them so they can be found.
* If you resubmit, please make sure to attach the file again. Your latest submission before the due date will be graded.
* **There is NO extended deadline for this assignment.**