­­­­

**Faculty of Computer and Information Sciences**

**Ain Shams University**

**Third Year – First Semester**

**2022 - 2023**

**FOS KERNEL PROJECT**

**Milestone 2 Appendix**

**Operating Systems**

Contents

[APPENDIX I: Lists Helper Functions 3](#_Toc117726017)

[Iterate on ALL Elements of a Specific List 3](#_Toc117726018)

[Get the size of any list 3](#_Toc117726019)

[Get the last element in a list 3](#_Toc117726020)

[Get the first element in a list 4](#_Toc117726021)

[Get the previous element to another element in a list 4](#_Toc117726022)

[Get the next element to another element in a list 4](#_Toc117726023)

[Remove a specific element in a list 4](#_Toc117726024)

[Insert a new element at the BEGINNING of a list 4](#_Toc117726025)

[Insert a new element at the END of a list 5](#_Toc117726026)

[Insert a new element before another element in a list 5](#_Toc117726027)

[Insert a new element after another element in a list 5](#_Toc117726028)

[APPENDIX II: Command Prompt 6](#_Toc117726029)

[Print current kernel heap placement strategy (NEXT FIT, BEST FIT, …) 6](#_Toc117726030)

[Changing kernel heap placement strategy (NEXT FIT, BEST FIT, …) 6](#_Toc117726031)

[Basic Functions 7](#_Toc117726032)

[Other Helpers Functions 7](#_Toc117726033)

APPENDICES

# APPENDIX I: Lists Helper Functions

IMPORTANT: you should pass all the lists to the functions by reference

Put **&** before the name of the list

### Iterate on ALL Elements of a Specific List

#### Description:

Used to loop on all frames in the given list

#### Function declaration:

LIST\_FOREACH (Type\_inside\_list\* iterator, Linked\_List\* list)

#### Parameters:

list: pointer to the linked list to loop on its elements

iterator: pointer to the current element in the list

#### Example:

struct ELEMENTDataType \*element;

**LIST\_FOREACH**(element, **&**(ActiveList))

{

//write your code.

}

### Get the size of any list

#### Description:

Used to retrieve the current size of a given list

#### Function declaration:

int size = LIST\_SIZE(Linked\_List \* list)

#### Parameters:

list: pointer to the linked list

#### Example:

int size = LIST\_SIZE(**&**(curenv->ActiveList))

### Get the last element in a list

#### Description:

Used to retrieve the last element in a list

#### Function declaration:

#### Type\_inside\_list\* element = LIST\_LAST(Linked\_List \* list)

#### Parameters:

list: pointer to the linked list

### Get the first element in a list

#### Description:

Used to retrieve the first element in a list (what the head points to)

#### Function declaration:

#### Type\_inside\_list\* element = LIST\_FIRST(Linked\_List \* list)

#### Parameters:

list: pointer to the linked list

### Get the previous element to another element in a list

#### Description:

Used to retrieve the previous element to another in a list

#### Function declaration:

#### Type\_inside\_list\* element = LIST\_PREV(Type\_inside\_list\* element)

#### Parameters:

element: is the element to get its previous

### Get the next element to another element in a list

#### Description:

Used to retrieve the next element to another in a list

#### Function declaration:

#### Type\_inside\_list\* element = LIST\_NEXT(Type\_inside\_list\* element)

#### Parameters:

element: is the element to get its next

### Remove a specific element in a list

#### Description:

Used to remove an given element from a list

#### Function declaration:

#### LIST\_REMOVE(Linked\_List \* list, Type\_inside\_list\* element)

#### Parameters:

list: pointer to the linked list

element: is the element to be removed from the given list

### Insert a new element at the BEGINNING of a list

#### Description:

Used to insert a new element at the head of a list

#### Function declaration:

#### LIST\_INSERT\_HEAD(Linked\_List \* list, Type\_inside\_list\* element)

#### Parameters:

list: pointer to the linked list

element: the new element to be inserted at the head of list

### Insert a new element at the END of a list

#### Description:

Used to insert a new element at the tail of a list

#### Function declaration:

#### LIST\_INSERT\_TAIL(Linked\_List \* list, Type\_inside\_list\* element)

#### Parameters:

list: pointer to the linked list

element: the new element to be inserted at the tail of list

### Insert a new element before another element in a list

#### Description:

Used to insert a new element before another element in a list

#### Function declaration:

#### LIST\_INSERT\_BEFORE(Linked\_List \* list, Type\_inside\_list\* elementToInsertBefore, Type\_inside\_list\* element)

#### Parameters:

list: pointer to the linked list

elementToInsertBefore: the element to insert the new element before it

element***:*** the new element to be inserted

### Insert a new element after another element in a list

#### Description:

Used to insert a new element after another element in a list

#### Function declaration:

#### LIST\_INSERT\_BEFORE(Linked\_List \* list, Type\_inside\_list\* elementToInsertAfter, Type\_inside\_list\* element)

#### Parameters:

list: pointer to the linked list

elementToInsertAfter: the element to insert the new element after it

element***:*** the new element to be inserted

# APPENDIX II: Command Prompt

### Print current kernel heap placement strategy (NEXT FIT, BEST FIT, …)

#### Name: kheap?

#### Description:

Print the current KERNEL heap placement strategy (NEXT FIT, BEST FIT, …).

### Changing kernel heap placement strategy (NEXT FIT, BEST FIT, …)

#### Name: khnextfit (khbestfit, khfirstfit)

#### Description:

Set the current KERNEL heap placement strategy to NEXT FIT (BEST FIT, …).

# Basic Functions

The basic **memory manager functions** that you may need to use are defined in “kern/mem/memory\_manager.c” file:

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| allocate\_frame | Used to allocate a free frame from the free frame list |
| free\_frame | Used to free a frame by adding it to free frame list |
| map\_frame | Used to map a single page with a given virtual address into a given allocated frame, simply by setting the directory and page table entries |
| get\_page\_table | Used by “map\_frame” to get a pointer to the page table if exist |
| unmap\_frame | Used to un-map a frame at the given virtual address, simply by clearing the page table entry |
| get\_frame\_info | Used to get both the page table and the frame of the given virtual address |

# Other Helpers Functions

There are some **helper functions** that we may need to use them in the rest of the course:

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Defined in…** |
| PDX (uint32 virtual address) | Gets the page directory index in the given virtual address (10 bits from 22 – 31). | Inc/mmu.h |
| PTX (uint32 virtual address) | Gets the page table index in the given virtual address (10 bits from 12 – 21). | Inc/mmu.h |
| ROUNDUP  (uint32 value, uint32 align) | Rounds a given “value” to the nearest upper value that is divisible by “align”. | Inc/types.h |
| ROUNDDOWN  (uint32 value, uint32 align) | Rounds a given “value” to the nearest lower value that is divisible by “align”. | Inc/types.h |
| tlb\_invalidate  (uint32\* page\_directory, uint32 virtual address) | Refresh the cache memory (TLB) to remove the given virtual address from it. | Kern/mem/ memory\_manager.c |
| isKHeapPlacementStrategyFIRSTFIT() …] | Check which strategy is currently selected using the given functions. | Kern/mem/kheap.h |