­­­­

**Faculty of Computer and Information Sciences**

**Ain Shams University**

**Third Year – First Semester**

**2022 - 2023**

**FOS KERNEL PROJECT**

**Milestone 1**

**Operating Systems**

Contents

[APPENDIX I: String Helper Functions 3](#_Toc116829310)

[Convert from string to integer 3](#_Toc116829311)

[String split 3](#_Toc116829316)

[String Length 3](#_Toc116829321)

[String Comparison 4](#_Toc116829326)

[Find Character in String 4](#_Toc116829331)

[APPENDIX II: Lists Helper Functions 5](#_Toc116829335)

[Iterate on ALL Elements of a Specific List 5](#_Toc116829336)

[Get the size of any list 5](#_Toc116829337)

[Get the last element in a list 5](#_Toc116829338)

[Get the first element in a list 6](#_Toc116829339)

[Get the previous element to another element in a list 6](#_Toc116829340)

[Get the next element to another element in a list 6](#_Toc116829341)

[Remove a specific element in a list 6](#_Toc116829342)

[Insert a new element at the BEGINNING of a list 7](#_Toc116829343)

[Insert a new element at the END of a list 7](#_Toc116829344)

APPENDICES

# APPENDIX I: String Helper Functions

**Convert from string to integer**

**Function prototype:**

*long* **strtol**(**const** *char* **\***s, *char* **\*\***endptr, *int* base)

**Arguments:**

* S: string to be converted
* Endptr: if you want to return a pointer to the last character after finishing the conversion (set it to NULL)
* Base: number system to be used for conversion (10 for decimal, 16 for hexadecimal…)

**Return:**

* Long containing the integer value

**Source file:**

* String.h - String.c

**String split**

**Function prototype:**

*int* **strsplit**(*char* **\***string, *char* **\***SPLIT\_CHARS, *char* **\*\***argv, *int* **\*** argc)

**Arguments:**

* string: string to be split
* SPLIT\_CHARS: splitting characters
* argv: array of strings after splitting
* argc: number of split strings (i.e. size of the argv array)

**Return:**

* 1 if succeed, 0 otherwise

**Source file:**

* String.h - String.c

**String Length**

**Function prototype:**

*int* **strlen**(**const** *char* **\***s)

**Arguments:**

* s: string to get its length

**Return:**

* length (# of characters) of the given string

**Source file:**

* String.h - String.c

**String Comparison**

**Function prototype:**

*int* **strncmp**(**const** *char* **\***p, **const** *char* **\***q, uint32 n)

Compare the first "n" characters from string "p" with corresponding characters in string "q"

**Arguments:**

* p: first string to be compared
* q: second string to compare with it
* n: number of characters to be compared

**Return:**

* 0 if identical match, +ve/-ve value otherwise

**Source file:**

* String.h - String.c

**Find Character in String**

**Function prototype:**

*char* **\* strfind**(**const** *char* **\***s, *char* c)

Return a pointer to the first occurrence of 'c' in 's', or a pointer to the string-ending null character if the string has no 'c'

**Arguments:**

* s: string
* c: character to search with it

**Source file:**

* String.h - String.c

# APPENDIX II: 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