

LinkedList

Generated by Doxygen 1.9.8

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 linkedlist.h File Reference	3
2.1.1 Typedef Documentation	4
2.1.1.1 LinkedList	4
2.1.2 Function Documentation	4
2.1.2.1 ll_add()	4
2.1.2.2 ll_create()	4
2.1.2.3 ll_destroy()	5
2.1.2.4 ll_get()	5
2.1.2.5 ll_length()	5
2.1.2.6 ll_peek()	6
2.1.2.7 ll_peek_last()	6
2.1.2.8 ll_poll()	6
2.1.2.9 ll_pop()	7
2.1.2.10 ll_push()	7
2.1.2.11 ll_remove()	7
2.2 linkedlist.h	8
Index	9

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

linkedList.h	3
--	---

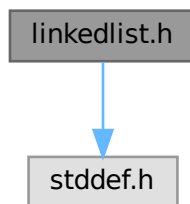
Chapter 2

File Documentation

2.1 linkedlist.h File Reference

```
#include <stddef.h>
```

Include dependency graph for linkedlist.h:



Typedefs

- typedef struct [LinkedList](#) * [LinkedList](#)

Functions

- [LinkedList ll_create](#) (size_t value_size)
Creates a new LinkedList.
- int [ll_push](#) ([LinkedList ll](#), const void *data)
Pushes a new value on the Stack.
- int [ll_add](#) ([LinkedList ll](#), const void *data)
Adds a new value to the LinkedList.
- void * [ll_pop](#) ([LinkedList ll](#))
Pops the first value from the Stack. Asserts that the Stack is not empty.
- void * [ll_poll](#) ([LinkedList ll](#))
Polls the first value from the Stack. If the Stack is empty NULL is returned.

- void * **ll_remove** (LinkedList ll, size_t index)
Removes an item from the LinkedList.
- void * **ll_peek** (LinkedList ll)
Returns the first item on the Stack.
- void * **ll_peek_last** (LinkedList ll)
Returns the last item on the Stack.
- void * **ll_get** (LinkedList ll, size_t index)
Gets the item at the index.
- int **ll_length** (LinkedList ll)
Returns the length of the LinkedList.
- int **ll_destroy** (LinkedList ll)
Destroys the LinkedList.

2.1.1 Typedef Documentation

2.1.1.1 LinkedList

```
typedef struct LinkedList* LinkedList
```

2.1.2 Function Documentation

2.1.2.1 ll_add()

```
int ll_add (
    LinkedList ll,
    const void * data )
```

Adds a new value to the LinkedList.

Parameters

<i>ll</i>	The LinkedList
<i>data</i>	The data to be added

Returns

Success code

2.1.2.2 ll_create()

```
LinkedList ll_create (
    size_t value_size )
```

Creates a new LinkedList.

Parameters

<i>value_size</i>	The sizeof value of the value
-------------------	-------------------------------

Returns

LinkedList

2.1.2.3 ll_destroy()

```
int ll_destroy (  
    LinkedList ll )
```

Destroys the LinkedList.

Parameters

//	The LinkedList
----	----------------

Returns

Success code

2.1.2.4 ll_get()

```
void * ll_get (  
    LinkedList ll,  
    size_t index )
```

Gets the item at the index.

Parameters

//	The LinkedList
<i>index</i>	index of the value

Returns

The item

2.1.2.5 ll_length()

```
int ll_length (  
    LinkedList ll )
```

Returns the length of the LinkedList.

Parameters

//	The LinkedList
----	----------------

Returns

The length

2.1.2.6 ll_peek()

```
void * ll_peek (
    LinkedList ll )
```

Returns the first item on the Stack.

Parameters

//	The LinkedList
----	----------------

Returns

The first value

2.1.2.7 ll_peek_last()

```
void * ll_peek_last (
    LinkedList ll )
```

Returns the last item on the Stack.

Parameters

//	The LinkedList
----	----------------

Returns

The last value

2.1.2.8 ll_poll()

```
void * ll_poll (
    LinkedList ll )
```

Polls the first value from the Stack. If the Stack is empty NULL is returned.

Parameters

//	The LinkedList
----	----------------

Returns

The polled value

2.1.2.9 ll_pop()

```
void * ll_pop (  
    LinkedList ll )
```

Pops the first value from the Stack. Asserts that the Stack is not empty.

Parameters

//	The LinkedList
----	----------------

Returns

The popped value

2.1.2.10 ll_push()

```
int ll_push (  
    LinkedList ll,  
    const void * data )
```

Pushes a new value on the Stack.

Parameters

<i>data</i>	The data to be pushed
-------------	-----------------------

Returns

Success code

2.1.2.11 ll_remove()

```
void * ll_remove (  
    LinkedList ll,  
    size_t index )
```

Removes an item from the LinkedList.

Parameters

<i>ll</i>	The LinkedList
<i>index</i>	The index from which an item should be removed

Returns

The removed value

2.2 linkedlist.h

[Go to the documentation of this file.](#)

```
00001 //
00002 // Created by Emanuel on 07.09.2024.
00003 //
00004
00005 #ifndef LINKEDLIST_H
00006 #define LINKEDLIST_H
00007
00008 #include <stddef.h>
00009
00010
00011 typedef struct LinkedList *LinkedList;
00012
00013
00020 LinkedList ll_create(size_t value_size);
00021
00028 int ll_push(LinkedList ll, const void *data);
00029
00037 int ll_add(LinkedList ll, const void *data);
00038
00045 void *ll_pop(LinkedList ll);
00046
00053 void *ll_poll(LinkedList ll);
00054
00062 void *ll_remove(LinkedList ll, size_t index);
00063
00070 void *ll_peek(LinkedList ll);
00071
00078 void *ll_peek_last(LinkedList ll);
00079
00087 void *ll_get(LinkedList ll, size_t index);
00088
00095 int ll_length(LinkedList ll);
00096
00103 int ll_destroy(LinkedList ll);
00104
00105
00106 #endif //LINKEDLIST_H
```

Index

- LinkedList
 - [linkedlist.h](#), 4
- [linkedlist.h](#), 3
 - LinkedList, 4
 - [ll_add](#), 4
 - [ll_create](#), 4
 - [ll_destroy](#), 5
 - [ll_get](#), 5
 - [ll_length](#), 5
 - [ll_peek](#), 6
 - [ll_peek_last](#), 6
 - [ll_poll](#), 6
 - [ll_pop](#), 7
 - [ll_push](#), 7
 - [ll_remove](#), 7
- [ll_add](#)
 - [linkedlist.h](#), 4
- [ll_create](#)
 - [linkedlist.h](#), 4
- [ll_destroy](#)
 - [linkedlist.h](#), 5
- [ll_get](#)
 - [linkedlist.h](#), 5
- [ll_length](#)
 - [linkedlist.h](#), 5
- [ll_peek](#)
 - [linkedlist.h](#), 6
- [ll_peek_last](#)
 - [linkedlist.h](#), 6
- [ll_poll](#)
 - [linkedlist.h](#), 6
- [ll_pop](#)
 - [linkedlist.h](#), 7
- [ll_push](#)
 - [linkedlist.h](#), 7
- [ll_remove](#)
 - [linkedlist.h](#), 7