CircularDoublyLinkedList

Generated by Doxygen 1.8.11

Contents

•	Data	i Structure index	•						
	1.1	Data Structures	1						
2	File	Index	2						
	2.1	File List	2						
3	Data	a Structure Documentation							
	3.1	llist_t Struct Reference	2						
		3.1.1 Detailed Description	2						
		3.1.2 Field Documentation	3						
	3.2	Inode_t Struct Reference	3						
		3.2.1 Detailed Description	3						
		3.2.2 Field Documentation	4						
4	File	Documentation	4						
	4.1	llist.c File Reference	4						
		4.1.1 Function Documentation	5						
	4.2	llist.c	7						
	4.3	llist.h File Reference	8						
		4.3.1 Detailed Description	10						
		4.3.2 Typedef Documentation	10						
		4.3.3 Function Documentation	10						
	4.4	llist.h	12						
Index									
1	Da	ta Structure Index							
1.1	Da	ata Structuros							
1.1									
He	Here are the data structures with brief descriptions:								
	llist t								

Inode_t 3

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

llist.c

llist.h

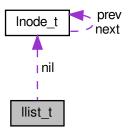
Circular doubly linked list definition and basic operations

3 Data Structure Documentation

3.1 Ilist_t Struct Reference

```
#include <llist.h>
```

Collaboration diagram for llist_t:



Data Fields

- size_t width
- Inode_t * nil
- int count

3.1.1 Detailed Description

Definition at line 22 of file Ilist.h.

3.1.2 Field Documentation

3.1.2.1 int count

count element amount

Definition at line 25 of file llist.h.

3.1.2.2 **Inode_t*** nil

sentinel (dummy node)

Definition at line 24 of file Ilist.h.

3.1.2.3 size_t width

element size (in bytes)

Definition at line 23 of file Ilist.h.

The documentation for this struct was generated from the following file:

· Ilist.h

3.2 Inode_t Struct Reference

#include <llist.h>

Collaboration diagram for Inode_t:



Data Fields

- struct Inode_t * prev
- void * data
- struct Inode_t * next

3.2.1 Detailed Description

Definition at line 16 of file Ilist.h.

3.2.2 Field Documentation

3.2.2.1 void* data

data pointer

Definition at line 18 of file Ilist.h.

3.2.2.2 struct Inode_t* next

next node

Definition at line 19 of file Ilist.h.

3.2.2.3 struct Inode_t* prev

previous node

Definition at line 17 of file Ilist.h.

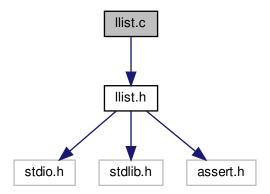
The documentation for this struct was generated from the following file:

• Ilist.h

4 File Documentation

4.1 Ilist.c File Reference

#include "llist.h"
Include dependency graph for llist.c:



4.1 Ilist.c File Reference 5

Functions

```
    Ilist_t * Ilist_create (size_t width)
```

- void llist_destruct (llist_t *I)
- Inode t * Ilist Isearch (Ilist t *I, int n)
- void llist_delete (llist_t *I, int n)
- Inode_t * Ilist_insert (Ilist_t *I, int n, void *e)
- void llist_int_print (llist_t *l)

4.1.1 Function Documentation

```
4.1.1.1 Ilist_t* llist_create ( size_t width )
```

Given the size of each element and the list size, create a list.

Parameters

widt	th	size of each element
max	_size	size of the list, max_size*width bytes will be reserved (definitively) for the list

Returns

a list initialized

Definition at line 10 of file llist.c.

4.1.1.2 void llist_delete ($llist_t * l$, int n)

Definition at line 60 of file llist.c.

```
00060
00061     if (1->count == 0) return;
00062     lnode_t* x = llist_lsearch(1, n);
00063     x->prev->next = x->next;
00064     x->next->prev = x->prev;
00065     free(x->data);
00066     free(x);
00067     1->count--;
00068 }
```

Here is the call graph for this function:



4.1.1.3 void llist_destruct (llist_t * I)

Free a list.

Parameters

q a list

Definition at line 27 of file llist.c.

4.1.1.4 Inode_t* llist_insert ($llist_t * l$, int n, void * e)

Definition at line 92 of file Ilist.c.

```
00092
00093
    lnode_t* x = l->count == 0 ? l->nil : llist_lsearch(l, n);
00094
    lnode_t* node = malloc(sizeof(lnode_t));
00095
    assert(node);
00096
    node->data = e;
00097
    llist_insert_ptr(x, node);
00098
    l->count++;
    return node;
00100 }
```

Here is the call graph for this function:



4.2 llist.c 7

```
4.1.1.5 void llist_int_print ( llist_t * I )
```

Print an int list

Definition at line 106 of file llist.c.

4.1.1.6 Inode t* llist_lsearch (llist_t * l, int n)

Definition at line 45 of file Ilist.c.

4.2 Ilist.c

```
00001 #include "llist.h"
00002
00010 llist_t* llist_create(size_t width) {
            llist_t* 1 = malloc(sizeof(llist_t));
00011
00012
            assert(1);
00013
            1->nil = malloc(sizeof(lnode_t));
            assert(1->nil);
1->nil->prev = 1->nil;
1->nil->next = 1->nil;
00014
00015
00016
00017
            1->width = width;
00018
            1->count = 0;
00019
            return 1;
00020 }
00021
00027 void llist_destruct(llist_t* 1) {
         lnode_t* x = 1->nil->next;
while(x != 1->nil) {
00029
00030
                 free(x->data);
00031
                 x = x->next;
00032
                 free (x->prev);
00033
00034
            free(1->nil);
00035
            free(1);
00036 }
00037
00038 /*
00039 ^{\star} Given an index n, do a linear search on a list 00040 ^{\star} \param 1 a list
00041 * \param n index
00042 * \return the node of index n
00043 */
00044
00045 lnode_t* llist_lsearch(llist_t* 1, int n) { 00046 assert (n >= 0 || n < 1->count);
            lnode_t* x = 1->ni1->next;
for(int i = 0; i < n; i++) {</pre>
00047
00048
00049
               x = x->next;
00050
00051
            return x;
00052 }
00053
00054 /*
```

```
00055 \star Delete the node of index n
00056 * \param 1 a list
00057 * \param n index
00058 */
00059
00060 void llist_delete(llist_t* 1, int n) {
00061 if (1->count == 0) return;

00062 lnode_t* x = llist_lsearch(1, n);
00063
            x->prev->next = x->next;
            x->next->prev = x->prev;
00064
00065
            free (x->data);
00066
            free(x);
00067
             1->count--;
00068 }
00069
00070 /*
00071 * Insert in the front of a given \ensuremath{\backslash}\ensuremath{\text{e}} node a node \ensuremath{\backslash}\ensuremath{\text{e}} x 00072 * \ensuremath{\backslash}\ensuremath{\text{param}} node the node to prepend
00073 \star \param x the node to insert
00074 */
00075
00076 static void llist_insert_ptr(lnode_t* node, lnode_t* x) {
00077
         lnode_t* pn = node->prev;
x->next = pn->next;
00078
            pn->next->prev = x;
00079
08000
            pn->next = x;
00081
             x->prev = pn;
00082 }
00083
00084 /*
00085 ^{\star} Given an index n, insert in the front of the node n 00086 ^{\star} \param 1 a list
00087 * \param n index
00088 * \param e element
00089 \, \, \return the new node of index n which be inserted 00090 \, \, \, ^{\prime}
00091
00095
             assert (node);
00096
            node->data = e;
00097
             llist_insert_ptr(x, node);
00098
             1->count++;
00099
             return node;
00100 }
00101
00106 void llist_int_print(llist_t* 1) {
00107     printf("%d nodes : nil<->", 1->count);
00108     lnode_t* x = 1->nil->next;
00109     for(int i = 0; i < 1->count; i++) {
00110
                 printf("[%d]<->", *((int*)x->data));
00111
                  x = x->next;
00112
             puts("nil");
00113
00114 }
```

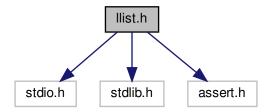
4.3 Ilist.h File Reference

Circular doubly linked list definition and basic operations.

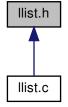
```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
```

4.3 Ilist.h File Reference 9

Include dependency graph for llist.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct Inode_t
- struct llist_t

Typedefs

- typedef struct Inode_t Inode_t
- typedef struct llist_t llist_t

Functions

- llist_t * llist_create (size_t width)
- void llist_destruct (llist_t *I)
- Inode_t * Ilist_insert (Ilist_t *I, int n, void *e)
- void llist_delete (llist_t *I, int n)
- void llist_int_print (llist_t *I)
- Inode_t * Ilist_Isearch (Ilist_t *I, int n)

4.3.1 Detailed Description

Circular doubly linked list definition and basic operations.

Author

Firmin MARTIN

Version

0.1

Date

01/01/2018

Definition in file Ilist.h.

- 4.3.2 Typedef Documentation
- 4.3.2.1 typedef struct llist_t llist_t
- 4.3.2.2 typedef struct Inode_t Inode_t
- 4.3.3 Function Documentation
- 4.3.3.1 Ilist_t* Ilist_create (size_t width)

Given the size of each element and the list size, create a list.

Parameters

	width	size of each element				
ſ	max_size	size of the list, max_size*width bytes will be reserved (definitively) for the list				

Returns

a list initialized

Definition at line 10 of file llist.c.

4.3 Ilist.h File Reference 11

```
4.3.3.2 void llist_delete ( llist_t * l, int n )
```

Definition at line 60 of file llist.c.

```
00060
00061    if (1->count == 0) return;
00062    lnode_t* x = llist_lsearch(1, n);
00063    x->prev->next = x->next;
00064    x->next->prev = x->prev;
00065    free(x->data);
00066    free(x);
00067    l->count--;
00068 }
```

Here is the call graph for this function:



```
4.3.3.3 void llist_destruct ( llist_t * / )
```

Free a list.

Parameters

```
q a list
```

Definition at line 27 of file Ilist.c.

```
00027
          lnode_t* x = l->nil->next;
while(x != l->nil) {
00028
00029
           free(x->data);
00030
00031
               x = x->next;
00032
               free(x->prev);
00033
           free(l->nil);
00034
00035
           free(1);
00036 }
```

4.3.3.4 Inode_t* llist_insert (llist_t * l, int n, void * e)

Definition at line 92 of file llist.c.

```
00092
00093
    lnode_t* x = l->count == 0 ? l->nil : llist_lsearch(l, n);
00094
    lnode_t* node = malloc(sizeof(lnode_t));
00095
    assert(node);
00096
    node->data = e;
00097
    llist_insert_ptr(x, node);
00098
    l->count++;
    return node;
00100 }
```

Here is the call graph for this function:



```
4.3.3.5 void llist_int_print ( llist_t * / )
```

Print an int list

Definition at line 106 of file llist.c.

4.3.3.6 Inode_t* llist_lsearch (llist_t * l, int n)

Definition at line 45 of file Ilist.c.

4.4 Ilist.h

```
00001 #ifndef LLIST_H
00002 #define LLIST_H
00003
00004 #include <stdio.h>
00005 #include <stdlib.h>
00006 #include <assert.h>
00007
00016 typedef struct lnode_t {
00017
            struct lnode_t* prev;
            void* data;
00018
            struct lnode_t* next;
00020 } lnode_t;
00021
00022 typedef struct llist_t {
         size_t width;
lnode_t* nil;
00023
00024
00025
            int count;
00026 } llist_t;
00027
00028 llist_t* llist_create(size_t width);
00022 void llist_destruct(llist_t* 1);

00030 lnode_t* llist_insert(llist_t* 1, int n, void* e);

00031 void llist_delete(llist_t* 1, int n);

00032 void llist_int_print(llist_t* 1);
00033 lnode_t* llist_lsearch(llist_t* 1, int n);
00034
00035 #endif /* ifndef LLIST_H */
```

Index

coun	t	prev	
	llist_t, 3		Inode_t, 4
data		width	1
uaia	Inode_t, 4		llist_t, 3
llist.c	e, 4		
	llist_create, 5		
	llist_delete, 5		
	llist_destruct, 6		
	llist_insert, 6		
	llist_int_print, 6		
	llist_lsearch, 7		
llist.h			
	llist_create, 10		
	llist_delete, 10		
	llist_destruct, 11 llist_insert, 11		
	llist_int_print, 12		
	llist_Isearch, 12		
	llist_t, 10		
	Inode t, 10		
llist	create		
_	llist.c, 5		
	llist.h, 10		
llist_	delete		
	llist.c, 5		
	llist.h, 10		
	destruct		
	llist.c, 6		
	llist.h, 11		
llist_	nsert		
	llist.c, 6		
111:-4	llist.h, 11		
IIIST_	int_print		
	llist.c, 6		
	Ilist.h, 12 Isearch		
	llist.c, 7		
	llist.h, 12		
llist_			
	count, 3		
	llist.h, 10		
	nil, 3		
	width, 3		
Inode	e_t, 3		
	data, 4		
	llist.h, 10		
	next, 4		
	prev, 4		
next			
•	Inode_t, 4		
nil			
	llist_t, 3		