

LinkedList

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	2
2.1	File List	2
3	Data Structure Documentation	2
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	7
4.3.1	Detailed Description	9
4.3.2	Typedef Documentation	9
4.3.3	Function Documentation	9
4.4	llist.h	11
	Index	13

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

llist_t	2
-------------------------	----------

Inode_t	3
-------------------------	-------------------

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

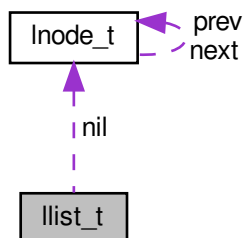
llist.c	4
llist.h	
Doubly circular linked list definition and basic operations	7

3 Data Structure Documentation

3.1 llist_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 [llist.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 [llist.h](#).

3.1.2.3 size_t width

element size (in bytes)

Definition at line 23 of file [llist.h](#).

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

- [llist.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 [llist.h](#).

3.2.2 Field Documentation

3.2.2.1 void* data

Definition at line 18 of file [llist.h](#).

3.2.2.2 struct Inode_t* next

Definition at line 19 of file [llist.h](#).

3.2.2.3 struct Inode_t* prev

Definition at line 17 of file [llist.h](#).

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

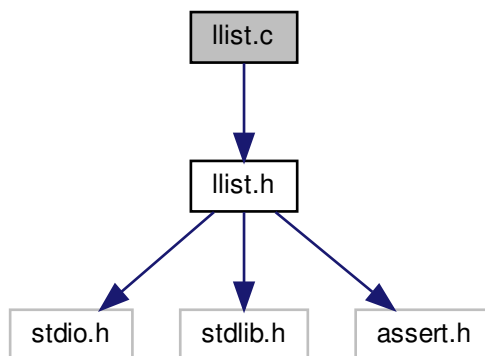
- [llist.h](#)

4 File Documentation

4.1 llist.c File Reference

```
#include "llist.h"
```

Include dependency graph for llist.c:



Functions

- [llist_t * llist_create](#) (size_t width)
- void [llist_destruct](#) (llist_t *l)
- [Inode_t * llist_lsearch](#) (llist_t *l, int n)
- void [llist_delete](#) (llist_t *l, int n)
- [Inode_t * llist_insert](#) (llist_t *l, int n, void *e)
- void [llist_int_print](#) (llist_t *l)

4.1.1 Function Documentation

4.1.1.1 llist_t* llist_create (size_t width)

Definition at line 3 of file [llist.c](#).

```
00003      {
00004          llist_t* l = malloc(sizeof(llist_t));
00005          assert(l);
00006          l->nil = malloc(sizeof(lnode_t));
00007          assert(l->nil);
00008          l->nil->prev = l->nil;
00009          l->nil->next = l->nil;
00010          l->width = width;
00011          l->count = 0;
00012          return l;
00013      }
```

4.1.1.2 void llist_delete (llist_t * l, int n)

Definition at line 35 of file [llist.c](#).

```
00035      {
00036          if (l->count == 0) return ;
00037          lnode_t* x = llist_lsearch(l, n);
00038          x->prev->next = x->next;
00039          x->next->prev = x->prev;
00040          free(x->data);
00041          free(x);
00042          l->count--;
00043      }
```

Here is the call graph for this function:



4.1.1.3 void llist_destruct (llist_t * l)

Definition at line 15 of file [llist.c](#).

```
00015      {
00016          lnode_t* x = l->nil->next;
00017          while(x != l->nil) {
00018              free(x->data);
00019              x = x->next;
00020              free(x->prev);
00021          }
00022          free(l->nil);
00023          free(l);
00024      }
```

4.1.1.4 `lnode_t* llist_insert (llist_t * l, int n, void * e)`

Definition at line 53 of file `llist.c`.

```

00053                                     {
00054     lnode_t* x = l->count == 0 ? l->nil : llist_lsearch(l, n);
00055     lnode_t* node = malloc(sizeof(lnode_t));
00056     assert(node);
00057     node->data = e;
00058     llist_insert_ptr(x, node);
00059     l->count++;
00060     return node;
00061 }
```

Here is the call graph for this function:



4.1.1.5 `void llist_int_print (llist_t * l)`

Definition at line 63 of file `llist.c`.

```

00063                                     {
00064     printf("%d nodes : nil<->", l->count);
00065     lnode_t* x = l->nil->next;
00066     for(int i = 0; i < l->count; i++) {
00067         printf("[%d]<->", *((int*)x->data));
00068         x = x->next;
00069     }
00070     puts("nil");
00071 }
```

4.1.1.6 `lnode_t* llist_lsearch (llist_t * l, int n)`

Definition at line 26 of file `llist.c`.

```

00026                                     {
00027     assert (n >= 0 || n < l->count) ;
00028     lnode_t* x = l->nil->next;
00029     for(int i = 0; i < n; i++) {
00030         x = x->next;
00031     }
00032     return x;
00033 }
```

4.2 llist.c

```

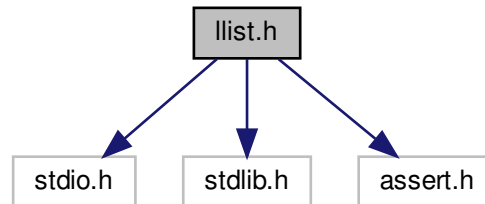
00001 #include "llist.h"
00002
00003 llist_t* llist_create(size_t width) {
00004     llist_t* l = malloc(sizeof(llist_t));
00005     assert(l);
00006     l->nil = malloc(sizeof(lnode_t));
00007     assert(l->nil);
00008     l->nil->prev = l->nil;
00009     l->nil->next = l->nil;
00010     l->width = width;
00011     l->count = 0;
00012     return l;
00013 }
00014
00015 void llist_destruct(llist_t* l) {
00016     lnode_t* x = l->nil->next;
00017     while(x != l->nil) {
00018         free(x->data);
00019         x = x->next;
00020         free(x->prev);
00021     }
00022     free(l->nil);
00023     free(l);
00024 }
00025
00026 lnode_t* llist_lsearch(llist_t* l, int n) {
00027     assert (n >= 0 || n < l->count) ;
00028     lnode_t* x = l->nil->next;
00029     for(int i = 0; i < n; i++) {
00030         x = x->next;
00031     }
00032     return x;
00033 }
00034
00035 void llist_delete(llist_t* l, int n) {
00036     if (l->count == 0) return ;
00037     lnode_t* x = llist_lsearch(l, n);
00038     x->prev->next = x->next;
00039     x->next->prev = x->prev;
00040     free(x->data);
00041     free(x);
00042     l->count--;
00043 }
00044
00045 static void llist_insert_ptr(lnode_t* node, lnode_t* x) {
00046     lnode_t* pn = node->prev;
00047     x->next = pn->next;
00048     pn->next->prev = x;
00049     pn->next = x;
00050     x->prev = pn;
00051 }
00052
00053 lnode_t* llist_insert(llist_t* l, int n, void* e) {
00054     lnode_t* x = l->count == 0 ? l->nil : llist_lsearch(l, n);
00055     lnode_t* node = malloc(sizeof(lnode_t));
00056     assert(node);
00057     node->data = e;
00058     llist_insert_ptr(x, node);
00059     l->count++;
00060     return node;
00061 }
00062
00063 void llist_int_print(llist_t* l) {
00064     printf("%d nodes : nil<->", l->count);
00065     lnode_t* x = l->nil->next;
00066     for(int i = 0; i < l->count; i++) {
00067         printf("[%d]<->", *((int*)x->data));
00068         x = x->next;
00069     }
00070     puts("nil");
00071 }

```

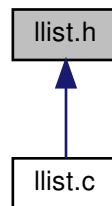
4.3 llist.h File Reference

Doubly circular linked list definition and basic operations.


```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
Include dependency graph for llist.h:
```



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



Data Structures

- struct [lnode_t](#)
- struct [llist_t](#)

Typedefs

- typedef struct [lnode_t](#) [lnode_t](#)
- typedef struct [llist_t](#) [llist_t](#)

Functions

- [llist_t *](#) [llist_create](#) (size_t width)
- void [llist_destruct](#) ([llist_t](#) *l)
- [lnode_t *](#) [llist_insert](#) ([llist_t](#) *l, int n, void *e)
- void [llist_delete](#) ([llist_t](#) *l, int n)
- void [llist_int_print](#) ([llist_t](#) *l)
- [lnode_t *](#) [llist_lsearch](#) ([llist_t](#) *l, int n)

4.3.1 Detailed Description

Doubly circular linked list definition and basic operations.

Author

Firmin MARTIN

Version

0.1

Date

01/01/2018

Definition in file [llist.h](#).

4.3.2 Typedef Documentation

4.3.2.1 typedef struct llist_t llist_t

4.3.2.2 typedef struct lnode_t lnode_t

4.3.3 Function Documentation

4.3.3.1 llist_t* llist_create (size_t width)

Definition at line 3 of file [llist.c](#).

```
00003 {
00004     llist_t* l = malloc(sizeof(llist_t));
00005     assert(l);
00006     l->nil = malloc(sizeof(lnode_t));
00007     assert(l->nil);
00008     l->nil->prev = l->nil;
00009     l->nil->next = l->nil;
00010     l->width = width;
00011     l->count = 0;
00012     return l;
00013 }
```

4.3.3.2 void llist_delete (llist_t* l, int n)

Definition at line 35 of file [llist.c](#).

```
00035 {
00036     if (l->count == 0) return ;
00037     lnode_t* x = llist_lsearch(l, n);
00038     x->prev->next = x->next;
00039     x->next->prev = x->prev;
00040     free(x->data);
00041     free(x);
00042     l->count--;
00043 }
```

Here is the call graph for this function:



4.3.3.3 void llist_destruct (llist_t * l)

Definition at line 15 of file llist.c.

```

00015      {
00016          lnode_t* x = l->nil->next;
00017          while(x != l->nil) {
00018              free(x->data);
00019              x = x->next;
00020              free(x->prev);
00021          }
00022          free(l->nil);
00023          free(l);
00024      }

```

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

Definition at line 53 of file llist.c.

```

00053      {
00054          lnode_t* x = l->count == 0 ? l->nil : llist_lsearch(l, n);
00055          lnode_t* node = malloc(sizeof(lnode_t));
00056          assert(node);
00057          node->data = e;
00058          llist_insert_ptr(x, node);
00059          l->count++;
00060          return node;
00061      }

```

Here is the call graph for this function:



4.3.3.5 void llist_int_print (llist_t * l)

Definition at line 63 of file llist.c.

```

00063      {
00064          printf("%d nodes : nil<->", l->count);
00065          lnode_t* x = l->nil->next;
00066          for(int i = 0; i < l->count; i++) {
00067              printf("[%d]<->", *((int*)x->data));
00068              x = x->next;
00069          }
00070          puts("nil");
00071      }

```

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

Definition at line 26 of file [llist.c](#).

```
00026 {
00027     assert (n >= 0 || n < l->count) ;
00028     lnode_t* x = l->nil->next;
00029     for(int i = 0; i < n; i++) {
00030         x = x->next;
00031     }
00032     return x;
00033 }
```

4.4 llist.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;
00018     void* data;
00019     struct lnode_t* next;
00020 } lnode_t;
00021
00022 typedef struct llist_t {
00023     size_t width;
00024     lnode_t* nil;
00025     int count;
00026 } llist_t;
00027
00028 llist_t* llist_create(size_t width);
00029 void llist_destruct(llist_t* l);
00030 lnode_t* llist_insert(llist_t* l, int n, void* e);
00031 void llist_delete(llist_t* l, int n);
00032 void llist_int_print(llist_t* l);
00033 lnode_t* llist_lsearch(llist_t* l, int n);
00034
00035 #endif /* ifndef LLIST_H */
```


Index

count

llist_t, 3

data

Inode_t, 4

llist.c, 4

llist_create, 5

llist_delete, 5

llist_destruct, 5

llist_insert, 5

llist_int_print, 6

llist_lsearch, 6

llist.h, 7

llist_create, 9

llist_delete, 9

llist_destruct, 9

llist_insert, 10

llist_int_print, 10

llist_lsearch, 10

llist_t, 9

Inode_t, 9

llist_create

llist.c, 5

llist.h, 9

llist_delete

llist.c, 5

llist.h, 9

llist_destruct

llist.c, 5

llist.h, 9

llist_insert

llist.c, 5

llist.h, 10

llist_int_print

llist.c, 6

llist.h, 10

llist_lsearch

llist.c, 6

llist.h, 10

llist_t, 2

count, 3

llist.h, 9

nil, 3

width, 3

Inode_t, 3

data, 4

llist.h, 9

next, 4

prev, 4

next

Inode_t, 4

nil

llist_t, 3

prev

Inode_t, 4

width

llist_t, 3