SinglyLinkedList

Generated by Doxygen 1.8.11

Contents

1	Data	Structi	ure Index	1		
	1.1	Data S	Structures	1		
2	File Index					
	2.1	File Lis	st	2		
3	Data	Structi	ure Documentation	2		
	3.1	llist_t S	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	Docum	entation	4		
	4.1	llist.c F	File Reference	4		
		4.1.1	Detailed Description	5		
		4.1.2	Function Documentation	5		
	4.2	llist.c .		7		
	4.3	llist.h F	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		
Inc	dex			13		

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Illist_t 2

Inode_t 3

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

llist.c

Singly linked list basic operations implementation

llist.h

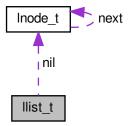
Singly linked list definition and basic operations

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 21 of file Ilist.h.

3.1.2 Field Documentation

3.1.2.1 int count

count element amount

Definition at line 24 of file Ilist.h.

sentinel (dummy node)

Definition at line 23 of file Ilist.h.

3.1.2.3 size_t width

element size (in bytes)

Definition at line 22 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

- 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 17 of file llist.h.

3.2.2.2 struct Inode_t* next

next node

Definition at line 18 of file Ilist.h.

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

· Ilist.h

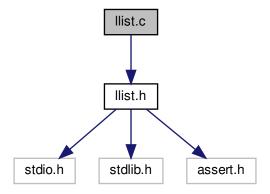
4 File Documentation

4.1 Ilist.c File Reference

Singly linked list basic operations implementation.

#include "llist.h"

Include dependency graph for llist.c:



Functions

- llist_t * llist_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 *I)

4.1 Ilist.c File Reference 5

4.1.1 Detailed Description

Singly linked list basic operations implementation.

Author

Firmin MARTIN

Version

0.1

Date

03/01/2018

Definition in file llist.c.

4.1.2 Function Documentation

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

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

Parameters

width	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 18 of file llist.c.

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

Definition at line 69 of file llist.c.

```
00069
00070    if (1->count == 0) return;
00071    /* get the previous node */
00072    lnode_t* x = llist_lsearch(l, n - 1);
00073    lnode_t* p = x->next;
00074    x->next = p->next;
00075    free(p->data);
00076    free(p);
00077    l->count--;
00078 }
```

Here is the call graph for this function:



4.1.2.3 void llist_destruct (llist_t * I)

Free a list.

Parameters

q a list

Definition at line 34 of file Ilist.c.

```
00034
           lnode_t* x = l->nil->next, *p;
while (x != l->nil) {
00035
00036
              p = x;
x = x->next;
00037
00038
00039
                free(p->data);
00040
                free(p);
00041
00042
            free(l->nil);
00043
            free(1);
00044 }
```

4.1.2.4 Inode_t* llist_insert (llist_t * I, int n, void * e)

Definition at line 99 of file llist.c.

```
00099
                                                                              {
             /* get the previous node */
lnode_t* x = llist_lsearch(1, n - 1);
lnode_t* node = malloc(sizeof(lnode_t));
00100
00101
00102
00103
              assert (node);
00104
             node->data = e;
00105
             llist_insert_ptr(x, node);
00106
             1->count++;
00107
             return node;
00108 }
```

4.2 Ilist.c 7

Here is the call graph for this function:



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

Print an int list

Definition at line 114 of file llist.c.

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

Definition at line 54 of file llist.c.

4.2 llist.c

```
00001 #include "llist.h"
00018 llist_t* llist_create(size_t width) {
00019
          llist_t* 1 = malloc(sizeof(llist_t));
00020
          assert(l);
          labstrict;
1->nil = malloc(sizeof(lnode_t));
assert(l->nil);
1->nil->next = 1->nil;
00021
00022
00023
00024
           l->width = width;
          1->count = 0;
00025
00026
           return 1;
00027 }
00028
00034 void llist_destruct(llist_t* 1) {
00035
        lnode_t * x = l->nil->next, *p;
00036
           while (x != 1->nil) {
             p = x;
x = x->next;
00037
00038
00039
               free (p->data);
00040
               free(p);
00041
           }
```

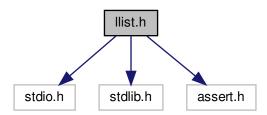
```
00042
           free(l->nil);
00043
           free(1);
00044 }
00045
00046 /*
00047 * Given an index n, do a linear search on a list, return the node. 00048 * If n == -1, return sentinel node.
00049
       * \param l a list
00050 * \param n index
00051 * \return the node of index n 00052 */
00053
00054 lnode_t* llist_lsearch(llist_t* 1, int n) {
00055
        assert (n \geq= -1 || n < 1->count) ;
            lnode_t* x = 1->nil;
for(int i = -1; i < n; i++) {</pre>
00056
00057
               x = x->next;
00058
00059
00060
            return x;
00061 }
00062
00063 /*
00064 * Delete the node of index n 00065 * \gamma la list
00066 * \param n index
00067 */
00068
00069 void llist_delete(llist_t* 1, int n) {
         if (1->count == 0) return ;
/* get the previous node */
00070
00071
00072
           lnode t*x = 1 list lsearch(1, n - 1);
00073
           lnode_t* p = x->next;
00074
           x \rightarrow next = p \rightarrow next;
00075
           free(p->data);
00076
            free(p);
00077
           1->count--:
00078 }
00079
00080 /*
00081 * Insert a node \e x after a given node \e node 00082 * \param node the node to prepend
00083 \, \star \param x the node to insert
00084 */
00085
00086 static void llist_insert_ptr(lnode_t* before, lnode_t* after) {
00087
         after->next = before->next;
00088
           before->next = after;
00089 }
00090
00091 /*
00092 \star Given an index n, insert in the front of the node n 00093 \star \param 1 a list
00094 * \param n index
00095 * \param e element 00096 * \return the new node of index n which be inserted
00097 */
00098
00099 lnode_t* llist_insert(llist_t* l, int n, void* e) {
00100 /* get the previous node */
           lnode_t* x = llist_lsearch(1, n - 1);
lnode_t* node = malloc(sizeof(lnode_t));
00101
00102
00103
           assert (node);
00104
           node->data = e;
00105
           llist_insert_ptr(x, node);
00106
            1->count++;
00107
            return node;
00108 }
00109
00114 void llist_int_print(llist_t* 1) {
         printf("%d nodes : nil->", l->count);
            lnode_t* x = 1->ni1->next;
for(int i = 0; i < 1->count; i++) {
    printf("[%d]->", *((int*)x->data));
00116
00117
00118
00119
                 x = x->next;
00120
00121
           puts("nil");
00122 }
```

4.3 Ilist.h File Reference

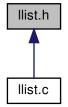
Singly linked list definition and basic operations.

4.3 Ilist.h File Reference 9

```
#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 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 (llist_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

Singly linked list definition and basic operations.

Author

Firmin MARTIN

Version

0.1

Date

03/01/2018

Definition in file Ilist.h.

- 4.3.2 Typedef Documentation
- 4.3.2.1 typedef struct Ilist_t Ilist_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
max_size	size of the list, max_size*width bytes will be reserved (definitively) for the list

Returns

a list initialized

Definition at line 18 of file Ilist.c.

4.3 Ilist.h File Reference 11

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

Definition at line 69 of file llist.c.

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 34 of file llist.c.

```
00034
00035
            lnode_t * x = 1->nil->next, *p;
            while (x != 1->nil) {
    p = x;
    x = x->next;
00036
00037
00038
                free(p->data);
00040
                free(p);
00041
00042
            free(l->nil);
00043
            free(1);
00044 }
```

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

Definition at line 99 of file Ilist.c.

```
00099
                                                                              {
00100
              /* get the previous node */
lnode_t* x = llist_lsearch(1, n - 1);
lnode_t* node = malloc(sizeof(lnode_t));
00101
00102
00103
              assert(node);
00104
             node->data = e;
00105
              llist_insert_ptr(x, node);
00106
             1->count++;
00107
             return node;
00108 }
```

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 114 of file llist.c.

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

Definition at line 54 of file Ilist.c.

```
00054
00055    assert (n >= -1 || n < 1->count);
00056    lnode_t* x = 1->nil;
00057    for(int i = -1; i < n; i++) {
        x = x->next;
00059    }
00060    return x;
00061 }
```

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 void* data;
00018 struct lnode
            struct lnode_t* next;
00019 } lnode_t;
00020
00021 typedef struct llist_t {
         size_t width;
lnode_t* nil;
00022
00023
00024
            int count;
00025 } llist_t;
00026
00027 llist_t* llist_create(size_t width);
00028 void llist_destruct(llist_t* 1);

00029 lnode_t* llist_insert(llist_t* 1, int n, void* e);

00030 void llist_delete(llist_t* 1, int n);

00031 void llist_int_print(llist_t* 1);
00032 lnode_t* llist_lsearch(llist_t* 1, int n);
00033
00034 #endif /* ifndef LLIST_H */
```

Index

```
count
     llist_t, 3
data
     Inode_t, 4
llist.c, 4
     llist_create, 5
     llist_delete, 5
     llist_destruct, 6
     llist_insert, 6
     llist_int_print, 7
     llist_lsearch, 7
llist.h, 8
     llist_create, 10
     llist_delete, 10
     llist_destruct, 11
     llist_insert, 11
     llist_int_print, 12
     llist_lsearch, 12
     llist_t, 10
     Inode_t, 10
llist_create
     llist.c, 5
     llist.h, 10
llist_delete
     llist.c, 5
     llist.h, 10
llist_destruct
     llist.c, 6
     llist.h, 11
llist insert
     llist.c, 6
     llist.h, 11
llist_int_print
     llist.c, 7
     llist.h, 12
llist_lsearch
     llist.c, 7
     llist.h, 12
llist_t, 2
     count, 3
     llist.h, 10
     nil, 3
     width, 3
Inode_t, 3
     data, 4
     llist.h, 10
     next, 4
next
     Inode_t, 4
nil
     llist_t, 3
width
     llist_t, 3
```