# Data Stuctures Library

Generated by Doxygen 1.8.4

Tue May 28 2013 22:49:09

# Contents

# Chapter 1

# Class Index

## 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 lnode Struct Reference

```
#include <sllist.h>
```

### Public Attributes

- void ∗ **data**
- struct lnode ∗ **next**

### 3.1.1 Detailed Description

The node structure.

The purpose of this structure is to actually hold the data or "payload" to be stored in the list. The nodes are connected sequentially, and thus each node requires a second field to store the address of the next node.

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

- include/sllist.h

## 3.2 sll_fix Struct Reference

### Public Attributes

- struct sllist ∗ **sllist**

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

- test/unit.c

## 3.3 sllist Struct Reference

```
#include <sllist.h>
```

### Public Attributes

- struct lnode ∗ **head**
- struct lnode ∗ **tail**
- struct lnode ∗ **current**
- int **size**

### 3.3.1 Detailed Description

The list structure.

Metadata is contained here. A pointer to the first and last nodes in the list allows for several operations to be performed more quickly. There is also another pointer-to-node member variable for storing the location of a "current" or active node that presumably will have operations performed on it. Finally there is a size variable containing the total number of nodes. Note that the first index of the list is considered index zero.

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

- include/sllist.h

# Chapter 4

# File Documentation

## 4.1 include/sllist.h File Reference

Stuctures and functions for a singly-linked list API.

### Classes

- struct lnode
- struct sllist

### Functions

- struct sllist ∗ sllist_create (void)
- void sllist_destroy (struct sllist ∗sllist)
- int sllist_push_front (struct sllist ∗sllist, void ∗data)
- int sllist_push_back (struct sllist ∗sllist, void ∗data)
- void ∗ sllist_pop_front (struct sllist ∗sllist)
- void ∗ sllist_pop_back (struct sllist ∗sllist)
- int sllist_step (struct sllist ∗sllist)
- void ∗ sllist_read_index (struct sllist ∗sllist, int index)
- int sllist_insert_after (struct sllist ∗sllist, int index, void ∗data)
- void ∗ sllist_extract_after (struct sllist ∗sllist, int index)

### 4.1.1 Detailed Description

Stuctures and functions for a singly-linked list API. The user provided with several different functions to manipulate lists and associated data.

### 4.1.2 Function Documentation

#### 4.1.2.1 struct sllist∗ sllist_create ( void )

Create a new list.

Returns a pointer to a new, empty list. If allocation fails, returns NULL.

#### 4.1.2.2 void sllist_destroy ( struct **sllist** ∗ *sllist* )

Destroy a list.

Frees the memory of the list struct and all associated nodes.

#### 4.1.2.3 void∗ sllist_extract_after ( struct **sllist** ∗ *sllist,* int *index* )

Extract a node after the node at the specified index.

Remove the specified node from the linked list, save a pointer to the data, free the node (but do not free the data itself), and return a pointer to the data so that it can be used. If the list is empty or the node doesn't exist in the list, returns NULL. Attempting to extract after the tail will also return NULL.

#### 4.1.2.4 int sllist_insert_after ( struct **sllist** ∗ *sllist,* int *index,* void ∗ *data* )

Insert a node after the node at the specified index.

Adds a node after the passed node. If allocation fails, returns -1. If the node doesn't exist in the list, returns 1. Otherwise, returns 0.

#### 4.1.2.5 void∗ sllist_pop_back ( struct **sllist** ∗ *sllist* )

Extract the last node.

Remove the last node from the linked list, save a pointer to the data, free the node (but do not free the data itself), and return a pointer to the data so that it can be used. If the list is empty, returns NULL.

#### 4.1.2.6 void∗ sllist_pop_front ( struct **sllist** ∗ *sllist* )

Extract the first node.

Remove the first node from the linked list, save a pointer to the data, free the node (but do not free the data itself), and return a pointer to the data so that it can be used. If the list is empty, returns NULL.

#### 4.1.2.7 int sllist_push_back ( struct **sllist** ∗ *sllist,* void ∗ *data* )

Append node to a list.

Adds a node to the end of the list. If allocation fails, returns -1, otherwise returns 0.

#### 4.1.2.8 int sllist_push_front ( struct **sllist** ∗ *sllist,* void ∗ *data* )

Prepend a node to the list:

Adds a node to the front of the list. If allocation fails, returns -1, otherwise returns 0.

#### 4.1.2.9 void∗ sllist_read_index ( struct **sllist** ∗ *sllist,* int *index* )

Access data by index.

Returns a pointer to the payload of the node at the location specified by the passed index value. The passed index value is interpreted as an offset from index zero, the first node of the list. Returns NULL if the list is empty or the index is out of range.

#### 4.1.2.10 int sllist_step ( struct **sllist** * *sllist* )

Step through a list.

Changes the current node to the node after the current node. Returns 1 if the current node is NULL.