MazBotV4 Helpers API

Most recent version

Bug tracker

Maz - programmer's diary

Contact: Mazziesaccount@gmail.com

****

Implementation of php's explode written in C
Written by Maz (2008)
Added Atomic operations for x86 architecture and
Linked list implementation.
Written by Maz (2009-2010)
http://maz-programmersdiary.blogspot.com/
You're free to use this piece of code.
You can also modify it freely, but if you
improve this, you must write the improved code
in comments at:
http://maz-programmersdiary.blogspot.com/
or at:
http://c-ohjelmoijanajatuksia.blogspot.com/
or mail the corrected version to me at
Mazziesaccount@gmail.com

Revision History:
- 0.0.6 15.08.2009/Maz Fixed atomic CAS
- 0.0.5 11.08.2009/Maz Added Cexplode_free_allButPieces
- 0.0.4 11.08.2009/Maz Added atomic ops and mbot_ll
- 0.0.3 31.07.2009/Maz Added Cexplode_concat (untested)
- 0.0.2 21.07.2009/Maz Some additions for better usability in MazBotV4
- 0.0.1 16.09.2008/Maz First Draft

****

Mon Feb 1 22:24:24 2010

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

CexplodeStrings (Struct for Cexplode object)	**2**
mbot_linkedList	**3**
MbotAtomic32 (Struct for 32bit wide integer type used in atomic operations)	**4**

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Data Structure Documentation

3.1 CexplodeStrings Struct Reference

Struct for Cexplode object.

```
#include <helpers.h>
```

Data Fields

- int amnt
- char ** strings
- char * separator
- int sepwasatend
- int startedWdelim
- int index

3.1.1 Detailed Description

Struct for Cexplode object.

3.1.2 Field Documentation

3.1.2.1 int amnt

3.1.2.2 int index

3.1.2.3 char∗ separator

3.1.2.4 int sepwasatend

3.1.2.5 int startedWdelim

3.1.2.6 char∗∗ strings

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

- helpers.h

3.2 mbot_linkedList Struct Reference

```
#include <helpers.h>
```

Data Fields

- struct mbot_linkedList ∗ head
- struct mbot_linkedList ∗ next
- struct mbot_linkedList ∗ prev
- void ∗ data

3.2.1 Field Documentation

3.2.1.1 void∗ data

3.2.1.2 struct mbot_linkedList∗ head `[read]`

3.2.1.3 struct mbot_linkedList∗ next [read]

3.2.1.4 struct mbot_linkedList∗ prev [read]

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

- helpers.h

3.3 MbotAtomic32 Struct Reference

Struct for 32bit wide integer type used in atomic operations.

`#include <helpers.h>`

Data Fields

- volatile unsigned int value
- sem_t sem

  If non x86 arch is used, these atomic ops are dummies using semaphore.

3.3.1 Detailed Description

Struct for 32bit wide integer type used in atomic operations.

3.3.2 Field Documentation

3.3.2.1 sem_t sem

If non x86 arch is used, these atomic ops are dummies using semaphore.

3.3.2.2 volatile unsigned int value

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

- helpers.h

4 File Documentation

4.1 helpers.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <semaphore.h>
```

Data Structures

- struct MbotAtomic32

  Struct for 32bit wide integer type used in atomic operations.

- struct CexplodeStrings

  Struct for Cexplode object.

- struct mbot_linkedList

Defines

- #define CEXPLODE_LAST_ITEM 0xFFFFFFFF

Enumerations

- enum ECexplodeRet { ECexplodeRet_InternalFailure = -666, ECexplodeRet_-InvalidParams = -667 }

  enumeration for Cexplodei's error return values

Functions

- int Cexplode_removeCurrent (CexplodeStrings ∗exp_obj)

  Removes the previously returned piece.

- char ∗ Cexplode_removeNth (int nro, CexplodeStrings ∗exp_obj)

  Removes Nth piece from cexplode Must not be called before calling Cexplode If removed item is last piece, the "sepwasatend" flag will be set true! Note, you can use special CEXPLODE_LAST_ITEM define to remove the last item.

- int Cexplode_getAmnt (CexplodeStrings exp_obj)

Get the amount of pieces in exploded object Must not be called before calling Cexplode.

- int Cexplode (const char ∗string, const char ∗delim, CexplodeStrings ∗exp_obj)

  Explodes string to pieces according to delimiter. Result is stored in exp_obj and can be retrieved using functions below The results of explosion are stored in same order as they occurred in initial string, eg. if string "1 2 3 4" would be exploded with space (" ") as delimiter, Cexplode_getfirst() would return 1, Cexplode_getNth() with n being 4, would return 4.

- int Cexplode_nextexists (CexplodeStrings exp_obj)

  Peeks if there's another result in exp_obj. Must not be called before calling Cexplode.

- char ∗ Cexplode_getNth (int index, CexplodeStrings ∗exp_obj)

  Retrieve's Nth exploded piece - first is first (index starts from 1, not from 0) Updates internal iterator, IE following call to Cexplode_getnext will retrieve index+1th piece.

- char ∗ Cexplode_getfirst (CexplodeStrings ∗exp_obj)

  Get's the first exploded piece. Same as Cexplode_getNth(1,∗exp_obj);.

- char ∗ Cexplode_getnext (CexplodeStrings ∗exp_obj)

  Get's next piece. Returns NULL if no more pieces are around.

- char ∗ Cexplode_getlast (CexplodeStrings ∗exp_obj)

  Gets last exploded piece.

- void Cexplode_free (CexplodeStrings exp_obj)

  Frees resources allocated by call to Cexplode() - BEWARE frees also splitted pieces.

- void Cexplode_free_allButPieces (CexplodeStrings exp_obj)

  Frees resources allocated by call to Cexplode() - does not free splitted pieces.

- size_t Cexplode_getlentilllast (CexplodeStrings exp_obj)

  Gets the amount of chars from the start of the original string to the beginning of last found delimiter.

- int Cexplode_sepwasatend (CexplodeStrings exp_obj)

  returns 1 if last chars in original string were the separator - else returns 0

- int Cexplode_concat (CexplodeStrings ∗first, CexplodeStrings ∗second)

  Concatenates two exp_objs into one. Modifies the first argument to contain new exp_obj. Does not modify second argument.

- int mbot_ltrim (char ∗text, char trimchar)

  removes trimchars from the beginning of a string.

- int mbot_rtrim (char ∗text, char trimchar)

  removes trailing trimchars from a string.

- int mbot_lrtrim (char ∗text, char trimchar)

  removes trailing trimchars as well as trimchars from the beginning of a string.

- int mbot_trimall (char ∗text, char trimchar)

  removes all trimchars from a string.

- MbotAtomic32 ∗ MbotAtomic32Init ()

  Creates 32bit atomic variable, compatible with mbot_atomic∗ operations.

- void MbotAtomic32Uninit (MbotAtomic32 ∗∗_this_)

  Uninitializes MbotAtomic32. This must not be called when it is possible someone is using the variable.

- unsigned int mbot_atomicGet (MbotAtomic32 ∗atomic)

  Get the value atomically.

- unsigned int mbot_atomicAdd (MbotAtomic32 ∗atomic, unsigned int addition)

  Increase value atomically - returns value before increment.

- unsigned int mbot_atomicDec (MbotAtomic32 ∗atomic, unsigned int decrement)

  Decrease value atomically - returns value before decrement.

- unsigned int mbot_atomicDecIfGreater (MbotAtomic32 ∗atomic, unsigned int decrement, unsigned int cmp)

  Decrease value atomically, if original value is greater than cmp. Returns original value. (If returnval<cmp, no decrement occurred.

- unsigned int mbot_atomicDecIfSmaller (MbotAtomic32 ∗atomic, unsigned int decrement, unsigned int cmp)

  Decrease value atomically, if original value is smaller than cmp. Returns original value. (If returnval>cmp, no decrement occurred.

- unsigned int mbot_atomicIncIfGreater (MbotAtomic32 ∗atomic, unsigned int decrement, unsigned int cmp)

  Increase value atomically, if original value is greater than cmp. Returns original value. (If returnval<cmp, no increment occurred.

- unsigned int mbot_atomicIncIfSmaller (MbotAtomic32 ∗atomic, unsigned int decrement, unsigned int cmp)

  Increase value atomically, if original value is smaller than cmp. Returns original value. (If returnval>cmp, no increment occurred.

- unsigned int mbot_atomicCAS (MbotAtomic32 ∗atomic, unsigned int old, unsigned int newval)
- mbot_linkedList ∗ mbot_ll_init ()

  Initializes linked list for use - returns ptr to list head.

- mbot_linkedList ∗ mbot_ll_get_prev (mbot_linkedList ∗_this)

  Gets previous list item. - returns previous item, or NULL if error occurred/first item given as param.

- mbot_linkedList ∗ mbot_ll_head_get (mbot_linkedList ∗_this)

  Get the head of the list Head can be used to maintain the location of empty list.

- mbot_linkedList ∗ mbot_ll_get_next (mbot_linkedList ∗_this)

  Get's next element - NULL if error occurred, or last element was provided as argument.

- mbot_linkedList ∗ mbot_ll_get_first (mbot_linkedList ∗_this)

  Get's the first list element - returns first element or NULL if no elements stored, or if an error occurred.

- mbot_linkedList ∗ mbot_ll_get_last (mbot_linkedList ∗_this)

  Gets the last element in list.

- mbot_linkedList ∗ mbot_ll_add (mbot_linkedList ∗_this, void ∗data)

  Adds item to list (data). Does not do a copy of data. Any list item (including head) can be used as _this.

- mbot_linkedList ∗ mbot_ll_release (mbot_linkedList ∗_this)

  removes given item from list - does not free memory.

- mbot_linkedList ∗ mbot_ll_safe_release (mbot_linkedList ∗_this, void ∗data)

  removes list item which holds data pointed by data. Any list item can be given in _this. Does not free memory. Returns removed list entry, and user must call free upon entry and stored data.

- void ∗ mbot_ll_dataGet (mbot_linkedList ∗_this)

  Gets data stored to an entry - entry and data are left untouched.

- void ∗ mbot_ll_dataSet (mbot_linkedList ∗_this, void ∗data)

    Sets data to an list,.

- mbot_linkedList ∗ mbot_ll_seek (mbot_linkedList ∗_this, void ∗data, size_t datasize)

    Searchs through the list and returns element in which the held data matches data specified in params.

- mbot_linkedList ∗ mbot_ll_copylist_wdata (mbot_linkedList ∗old, size_t itemsize)

    Copies given list and itemsize bytes of data from each container to new list, and returns a pointer to the copylist.

- void mbot_ll_destroy (mbot_linkedList ∗∗_this)

    Frees all entries from list, and destroys the list - does not free stored data. _this is NULLed upon return.

4.1.1 Define Documentation

4.1.1.1 #define CEXPLODE_LAST_ITEM 0xFFFFFFFF

4.1.2 Enumeration Type Documentation

4.1.2.1 enum ECexplodeRet

enumeration for Cexplodei's error return values

Enumerator:

ECexplodeRet_InternalFailure

ECexplodeRet_InvalidParams

4.1.3 Function Documentation

4.1.3.1 int Cexplode (const char ∗ *string*, const char ∗ *delim*, CexplodeStrings ∗ *exp_obj*)

Explodes string to pieces according to delimiter. Result is stored in exp_obj and can be retrieved using functions below The results of explosion are stored in same order as they occurred in initial string, eg. if string "1 2 3 4" would be exploded with space (" ") as delimiter, Cexplode_getfirst() would return 1, Cexplode_getNth() with n being 4, would return 4.

Parameters:

- ***const*** char ∗string pointer to C string being exploded
- ***const*** char ∗delim pointer to C string used as delimiter for cutting original string
- ***CexplodeStrings*** ∗exp_obj pointer to CexplodeStrings type object, which will be filled to contain results of explosion.

Returns:

amount of pieces - number smaller than 1 if an error occurs

See also:

CexplodeStrings, Cexplode_removeCurrent, Cexplode_removeNth, Cexplode_-getAmnt, Cexplode_nextexists, Cexplode_getNth, Cexplode_getfirst, Cexplode_-getnext, Cexplode_getlast, Cexplode_free, Cexplode_free_allButPieces, Cexplode_getlentilllast, Cexplode_sepwasatend, Cexplode_concat

4.1.3.2 int Cexplode_concat (CexplodeStrings ∗ *first*, CexplodeStrings ∗ *second*)

Concatenates two exp_objs into one. Modifies the first argument to contain new exp_-obj. Does not modify second argument.

Parameters:

- ***CexplodeStrings*** ∗first pointer to CexplodeStrings type object, filled by call to Cexplode() to be combined with another CexplodeStrings object. This will contain new CexplodeStrings object holding results for both of the original CexplodeStrings objects.
- ***CexplodeStrings*** ∗second ointer to CexplodeStrings type object, filled by call to Cexplode() to be combined with another CexplodeStrings object - this will not be modified during call.

Returns:

the amount of pieces in new exp_obj - negative number upon error.

Warning:

Must not be called before calling Cexplode for both first and second argument.

4.1.3.3 void Cexplode_free (CexplodeStrings *exp_obj*)

Frees resources allocated by call to Cexplode() - BEWARE frees also splitted pieces.

Parameters:

> ***CexplodeStrings*** exp_obj CexplodeStrings type object, filled by call to Cexplode()

Warning:

> Must not be called before calling Cexplode
> BEWARE frees also splitted pieces, in which the returned pointers by Cexplode_-get∗ points.

See also:

> Cexplode_free_allButPieces, Cexplode, Cexplode_getNth, Cexplode_getnext, Cexplode_getfirst, Cexplode_getlast

4.1.3.4 void Cexplode_free_allButPieces (CexplodeStrings *exp_obj*)

Frees resources allocated by call to Cexplode() - does not free splitted pieces.

Parameters:

> ***CexplodeStrings*** exp_obj CexplodeStrings type object, filled by call to Cexplode()

Warning:

> Must not be called before calling Cexplode

See also:

> Cexplode_free, Cexplode, Cexplode_getNth, Cexplode_getnext, Cexplode_-getfirst, Cexplode_getlast

4.1.3.5 int Cexplode_getAmnt (CexplodeStrings *exp_obj*)

Get the amount of pieces in exploded object Must not be called before calling Cexplode.

Parameters:

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

amount of exploded pieces stored in CexplodeStrings container

See also:

Cexplode

4.1.3.6 char∗ Cexplode_getfirst (CexplodeStrings ∗ *exp_obj*)

Get's the first exploded piece. Same as Cexplode_getNth(1,∗exp_obj);.

Parameters:

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

NULL on error, othervice a pointer to result stored in Cexplode object

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_getNth, Cexplode_getnext, Cexplode_getlast

4.1.3.7 char∗ Cexplode_getlast (CexplodeStrings ∗ *exp_obj*)

Gets last exploded piece.

Parameters:

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

NULL on error, othervice a pointer to result stored in Cexplode object

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_getNth, Cexplode_getnext, Cexplode_getfirst

4.1.3.8 size_t Cexplode_getlentilllast (CexplodeStrings *exp_obj*)

Gets the amount of chars from the start of the original string to the beginning of last found delimiter.

Parameters:

CexplodeStrings exp_obj CexplodeStrings type object, filled by call to Cexplode()

Returns:

amount of chars from the start of the original string to the beginning of last found delimiter

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_sepwasatend

4.1.3.9 char∗ Cexplode_getnext (CexplodeStrings ∗ *exp_obj*)

Get's next piece. Returns NULL if no more pieces are around.

Parameters:

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

NULL on error, othervice a pointer to result stored in Cexplode object

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_getNth, Cexplode_getfirst, Cexplode_getlast

4.1.3.10 char∗ Cexplode_getNth (int *index*, CexplodeStrings ∗ *exp_obj*)

Retrieve's Nth exploded piece - first is first (index starts from 1, not from 0) Updates internal iterator, IE following call to Cexplode_getnext will retrieve index+1th piece.

Parameters:

int index index number of result to be retrieved. first is first (index starts from 1, not from 0)

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

NULL on error, othervice a pointer to result stored in Cexplode object

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_getfirst, Cexplode_getnext, Cexplode_getlast, Cexplode_-getAmnt

4.1.3.11 int Cexplode_nextexists (CexplodeStrings *exp_obj*)

Peeks if there's another result in exp_obj. Must not be called before calling Cexplode.

Parameters:

CexplodeStrings exp_obj CexplodeStrings type object, filled by call to Cexplode()

Returns:

1 if next piece exists (Eg. if Cexplode_getnext et al. can be safely used), 0 if there's no next result in object.

See also:

Cexplode, Cexplode_getnext

4.1.3.12 int Cexplode_removeCurrent (CexplodeStrings ∗ *exp_obj*)

Removes the previously returned piece. Must not be called before calling Cexplode If removed item is last piece, the "sepwasatend" flag will be set true

Parameters:

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

0 at success, -1 at failure

See also:

Cexplode, Cexplode_removeNth, Cexplode_getAmnt, Cexplode_nextexists

4.1.3.13 char∗ Cexplode_removeNth (int *nro*, CexplodeStrings ∗ *exp_obj*)

Removes Nth piece from cexplode Must not be called before calling Cexplode If removed item is last piece, the "sepwasatend" flag will be set true! Note, you can use special CEXPLODE_LAST_ITEM define to remove the last item.

Parameters:

int nro number of exploded piece to be removed from the CexplodeStrings containing results

CexplodeStrings ∗exp_obj pointer to CexplodeStrings type object, filled by call to Cexplode()

Returns:

ptr to removed string

See also:

Cexplode, Cexplode_removeCurrent, Cexplode_getAmnt, Cexplode_nextexists

4.1.3.14 int Cexplode_sepwasatend (CexplodeStrings *exp_obj*)

returns 1 if last chars in original string were the separator - else returns 0

Parameters:

CexplodeStrings exp_obj CexplodeStrings type object, filled by call to Cexplode()

Returns:

1 if last chars in original string were the separator - else returns 0

Warning:

Must not be called before calling Cexplode

See also:

Cexplode, Cexplode_getlentilllast

4.1.3.15 unsigned int mbot_atomicAdd (MbotAtomic32 ∗ *atomic*, unsigned int *addition*)

Increase value atomically - returns value before increment.

Warning:

If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.16 unsigned int mbot_atomicCAS (MbotAtomic32 ∗ *atomic*, unsigned int *old*, unsigned int *newval*)

4.1.3.17 unsigned int mbot_atomicDec (MbotAtomic32 ∗ *atomic*, unsigned int *decrement*)

Decrease value atomically - returns value before decrement.

Warning:

> If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.18 unsigned int mbot_atomicDecIfGreater (MbotAtomic32 ∗ *atomic*, unsigned int *decrement*, unsigned int *cmp*)

Decrease value atomically, if original value is greater than cmp. Returns original value. (If returnval<cmp, no decrement occurred.

Warning:

> If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.19 unsigned int mbot_atomicDecIfSmaller (MbotAtomic32 ∗ *atomic*, unsigned int *decrement*, unsigned int *cmp*)

Decrease value atomically, if original value is smaller than cmp. Returns original value. (If returnval>cmp, no decrement occurred.

Warning:

> If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.20 unsigned int mbot_atomicGet (MbotAtomic32 ∗ *atomic*)

Get the value atomically.

Warning:

If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.21 unsigned int mbot_atomicIncIfGreater (MbotAtomic32 ∗ *atomic*, unsigned int *decrement*, unsigned int *cmp*)

Increase value atomically, if original value is greater than cmp. Returns original value. (If returnval<cmp, no increment occurred.

Warning:

If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.22 unsigned int mbot_atomicIncIfSmaller (MbotAtomic32 ∗ *atomic*, unsigned int *decrement*, unsigned int *cmp*)

Increase value atomically, if original value is smaller than cmp. Returns original value. (If returnval>cmp, no increment occurred.

Warning:

If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

4.1.3.23 mbot_linkedList∗ mbot_ll_add (mbot_linkedList ∗ *_this*, void ∗ *data*)

Adds item to list (data). Does not do a copy of data. Any list item (including head) can be used as _this.

Returns:

list entry corresponding to stored data

4.1.3.24 mbot_linkedList∗ mbot_ll_copylist_wdata (mbot_linkedList ∗ *old*, size_t *itemsize*)

Copies given list and itemsize bytes of data from each container to new list, and returns a pointer to the copylist.

Returns:

a pointer to the copylist and NULL on error

Warning:

This assumes that each "container" in list holds at least itemsize bytes of data - and copies exactly itemsize bytes.
Usable really only for lists which hold fixed size items!

4.1.3.25 void∗ mbot_ll_dataGet (mbot_linkedList ∗ *_this*)

Gets data stored to an entry - entry and data are left untouched.

4.1.3.26 void∗ mbot_ll_dataSet (mbot_linkedList ∗ *_this*, void ∗ *data*)

Sets data to an list,.

Returns:

previous data

Warning:

- this should be avoided. Malicious use may corrupt the list!

4.1.3.27 void mbot_ll_destroy (mbot_linkedList ∗∗ *_this*)

Frees all entries from list, and destroys the list - does not free stored data. _this is NULLed upon return.

4.1.3.28 mbot_linkedList∗ mbot_ll_get_first (mbot_linkedList ∗ *_this*)

Get's the first list element - returns first element or NULL if no elements stored, or if an error occurred.

4.1.3.29 mbot_linkedList∗ mbot_ll_get_last (mbot_linkedList ∗ *_this*)

Gets the last element in list.

4.1.3.30 mbot_linkedList∗ mbot_ll_get_next (mbot_linkedList ∗ *_this*)

Get's next element - NULL if error occurred, or last element was provided as argument.

4.1.3.31 mbot_linkedList∗ mbot_ll_get_prev (mbot_linkedList ∗ *_this*)

Gets previous list item. - returns previous item, or NULL if error occurred/first item given as param.

4.1.3.32 mbot_linkedList∗ mbot_ll_head_get (mbot_linkedList ∗ *_this*)

Get the head of the list Head can be used to maintain the location of empty list.

Returns:

the head, and NULL on error

Warning:

HEAD IS NOT SUPPOSED TO BE USED AS STORING ELEMENT!

4.1.3.33 mbot_linkedList∗ mbot_ll_init ()

Initializes linked list for use - returns ptr to list head.

4.1.3.34 mbot_linkedList∗ mbot_ll_release (mbot_linkedList ∗ *_this*)

removes given item from list - does not free memory.

Returns:

removed list entry, and user must call free upon entry and stored data.

4.1.3.35 mbot_linkedList∗ mbot_ll_safe_release (mbot_linkedList ∗ *_this*, void ∗ *data*)

removes list item which holds data pointed by data. Any list item can be given in _this. Does not free memory. Returns removed list entry, and user must call free upon entry and stored data.

Returns:

removed list entry

4.1.3.36 mbot_linkedList∗ mbot_ll_seek (mbot_linkedList ∗ *_this*, void ∗ *data*, size_t *datasize*)

Searchs through the list and returns element in which the held data matches data specified in params.

Warning:

, all elements must contain at least as much data as specified in size_t datasize!

4.1.3.37 int mbot_lrtrim (char ∗ *text*, char *trimchar*)

removes trailing trimchars as well as trimchars from the beginning of a string.

Returns:

number of characters removed

4.1.3.38 int mbot_ltrim (char ∗ *text*, char *trimchar*)

removes trimchars from the beginning of a string.

Returns:

number of characters removed

4.1.3.39 int mbot_rtrim (char ∗ *text*, char *trimchar*)

removes trailing trimchars from a string.

Returns:

number of characters removed

4.1.3.40 int mbot_trimall (char ∗ *text*, char *trimchar*)

removes all trimchars from a string.

Returns:

number of characters removed

4.1.3.41 MbotAtomic32∗ MbotAtomic32Init ()

Creates 32bit atomic variable, compatible with mbot_atomic∗ operations.

4.1.3.42 void MbotAtomic32Uninit (MbotAtomic32 ∗∗ *_this_*)

Uninitializes MbotAtomic32. This must not be called when it is possible someone is using the variable.

Warning:

If non x86 arch is used, these atomic ops are ineffective dummies using a huge semaphore (provided only for compatibility). On x86 arch compile with define ARCH_x86

Index