The cstring API

Macros

#define **cstring_string_type**(type)
cstring_string_type - The string type used in this library.

#define **cstring**(type)
cstring - Syntactic sugar to retrieve a string type.

#define **cstring_iterator**(type)
cstring_iterator - The iterator type used for a string.

#define **cstring_literal**(name, type, lit)
cstring_literal - Generate a cstring object with static duration.

#define **cstring_init**(name, type)
cstring_init - Allocate a new cstring with zero length.

#define **cstring_assign**(str, ptr, count)
cstring_assign - Assign a string to a cstring.

#define **cstring_free**(str)
cstring_free - Free all memory associated with the cstring and set it to NULL.

#define **cstring_at**(str, pos)
cstring_at - Return the character at position pos in the cstring.

#define **cstring_front**(str)
cstring_front - Return the first character in the cstring.

#define **cstring_back**(str)
cstring_back - Return the last character in the cstring.

#define **cstring_begin**(str)
cstring_begin - Return an iterator to first character of the string.

#define **cstring_end**(str)
cstring_end - Return an iterator to one past the last character of the string.

#define **cstring_empty**(str)
cstring_empty - Return 1 if the string is empty.

#define **cstring_size**(str)
cstring_size - Get the current length of the string.

#define **cstring_length**(str)
cstring_length - Get the current length of the string.

#define **cstring_max_size**(type)
cstring_max_size - Get the maximum number of elements a string of the specified character type is able to hold.

#define **cstring_reserve**(str, n)
cstring_reserve - Request that the string capacity be at least enough to contain n characters.

#define **cstring_capacity**(str)

	cstring_capacity - Get the current capacity of the string.
#define	**cstring_shrink_to_fit**(str) cstring_shrink_to_fit - Request the container to reduce its capacity to fit its size.
#define	**cstring_unsafe_set_size**(str, size) cstring_unsafe_set_size - Set the size property to the specified value and add the string terminator accordingly.
#define	**cstring_clear**(str) cstring_clear - Erase all of the characters in the string.
#define	**cstring_insert**(str, pos, ptr, count) cstring_insert - Insert a string at position `pos` into the cstring.
#define	**cstring_erase**(str, pos, n) cstring_erase - Remove the characters beginning at offset `pos` from the cstring.
#define	**cstring_push_back**(str, value) cstring_push_back - Add a character to the end of the string.
#define	**cstring_pop_back**(str) cstring_pop_back - Remove the last character from the cstring.
#define	**cstring_append**(str, ptr, count) cstring_append - Append a string at the end of the cstring.
#define	**cstring_replace**(str, pos, n, ptr, count) cstring_replace - Replace a substring beginning at position `pos` with another string.
#define	**cstring_copy**(from, to) cstring_copy - Copy a cstring.
#define	**cstring_resize**(str, count, value) cstring_resize - Resize the container to contain `count` characters.
#define	**cstring_swap**(str, other) cstring_swap - Exchange the content of the cstring by the content of another cstring of the same type.
#define	**cstring_trim**(str, value, mode) cstring_trim - Remove contiguous occurrences of the specified character from the begin and/or the end of a cstring.
#define	**cstring_fix**(str, length, value, mode) cstring_fix - Update the cstring to a fixed length by either padding or shortening.
#define	**cstring_reverse**(str) cstring_reverse - Reverse the character order in the cstring.
#define	**cstring_find**(str, pos, ptr, count, ret_offset) cstring_find - Find the first occurrence of the given substring.
#define	**cstring_rfind**(str, pos, ptr, count, ret_offset) cstring_rfind - Find the last occurrence of the given substring.
#define	**cstring_find_first_of**(str, pos, ptr, count, ret_offset) cstring_find_first_of - Find the first character equal to one of the characters in the given character sequence.

#define	**cstring_find_first_not_of**(str, pos, ptr, count, ret_offset) cstring_find_first_not_of - Find the first character equal to none of the characters in the given character sequence.
#define	**cstring_find_last_of**(str, pos, ptr, count, ret_offset) cstring_find_last_of - Find the last character equal to one of the characters in the given character sequence.
#define	**cstring_find_last_not_of**(str, pos, ptr, count, ret_offset) cstring_find_last_not_of - Find the last character equal to none of the characters in the given character sequence.
#define	**cstring_compare**(str1, str2, ret_order) cstring_compare - Lexicographically compare two strings.
#define	**cstring_starts_with**(str, ptr, count, ret_found) cstring_starts_with - Check if the string begins with the given prefix.
#define	**cstring_ends_with**(str, ptr, count, ret_found) cstring_ends_with - Check if the string ends with the given suffix.
#define	**cstring_contains**(str, ptr, count, ret_found) cstring_contains - Check if the string contains the given substring.
#define	**cstring_substring**(from, pos, n, to) cstring_substring - Copy a part of a string.

Detailed Description

Macro Definition Documentation

◆ cstring

#define cstring (type)

cstring - Syntactic sugar to retrieve a string type.

Parameters

type - The character type of the string to act on.

◆ cstring_append

```
#define cstring_append ( str,
                         ptr,
                         count )
```

cstring_append - Append a string at the end of the cstring.

Also see `cstring_init()`, `cstring_assign()`, `cstring_reserve()`, `cstring_push_back()`, `cstring_resize()`.

Parameters

str - The cstring. Can be a NULL string.

ptr - Pointer to the first character appended to the cstring.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_assign

```
#define cstring_assign ( str,
                         ptr,
                         count )
```

cstring_assign - Assign a string to a cstring.

Also see `cstring_init()`, `cstring_reserve()`, `cstring_push_back()`, `cstring_append()`, `cstring_resize()`.

Parameters

str - The cstring. Can be a NULL string.
If `str` refers to an existing cstring, the old content is overwritten.

ptr - Pointer to the first character assigned to the cstring.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_at

```
#define cstring_at ( str,
                     pos )
```

cstring_at - Return the character at position pos in the cstring.

Parameters

str - The cstring.

pos - Position of a character in the string.

Returns

The ASCII value (as `int`) of the character at the specified position in the string. If the macro fails, -1 is returned.

◆ cstring_back

```
#define cstring_back ( str )
```

cstring_back - Return the last character in the cstring.

Parameters

str - The cstring.

Returns

The ASCII value (as `int`) of the last character in the string. If the macro fails, -1 is returned.

◆ cstring_begin

```
#define cstring_begin ( str )
```

cstring_begin - Return an iterator to first character of the string.

Parameters

str - The cstring.

Returns

A pointer to the first character (or NULL).

◆ cstring_capacity

#define cstring_capacity (str)

cstring_capacity - Get the current capacity of the string.

Parameters

str - The cstring. Can be a NULL string.

Returns

The capacity as a `size_t`. Zero if `str` is NULL.

◆ cstring_clear

#define cstring_clear (str)

cstring_clear - Erase all of the characters in the string.

Parameters

str - The cstring.

Returns

void

◆ cstring_compare

#define cstring_compare (str1,
str2,
ret_order)

cstring_compare - Lexicographically compare two strings.

Parameters

str1 - The first cstring.

str2 - The second cstring.

ret_order - Variable of type `int` that receives the result of the comparison.
-1 if `str1` appears before `str2`
0 if both strings compare equivalent
1 if `str1` appears after `str2`

Returns

void

◆ cstring_contains

```
#define cstring_contains ( str,
                           ptr,
                           count,
                           ret_found )
```

cstring_contains - Check if the string contains the given substring.

Parameters

str - The cstring.

ptr - Pointer to the first character of the substring.

count - Length of the substring.

ret_found - Variable of type `int` that receives the result of the check.
0 if the substring was not found
1 if the substring was found

Returns

void

◆ cstring_copy

```
#define cstring_copy ( from,
                       to )
```

cstring_copy - Copy a cstring.

Parameters

from - The original cstring.

to - Destination to which the cstring is copied. Can be a NULL string.
If `to` refers to an existing cstring, the old content is overwritten.

Returns

void

◆ cstring_empty

#define cstring_empty (str)

cstring_empty - Return 1 if the string is empty.

Parameters

str - The cstring. Can be a NULL string.

Returns

1 if `str` is NULL or empty, 0 if non-empty.

◆ cstring_end

#define cstring_end (str)

cstring_end - Return an iterator to one past the last character of the string.

Parameters

str - The cstring.

Returns

A pointer to one past the last character (or NULL).

◆ cstring_ends_with

#define cstring_ends_with (str,
ptr,
count,
ret_found)

cstring_ends_with - Check if the string ends with the given suffix.

Parameters

str - The cstring.

ptr - Pointer to the first character of the suffix.

count - Length of the suffix.

ret_found - Variable of type `int` that receives the result of the check.
0 if the suffix was not found
1 if the suffix was found

Returns

void

◆ cstring_erase

```
#define cstring_erase ( str,
                        pos,
                        n )
```

cstring_erase - Remove the characters beginning at offset `pos` from the cstring.

Parameters

str - The cstring.

pos - Offset of the first character erased from the cstring.

n - Number of consecutive characters to be erased.

Returns

void

◆ cstring_find

```
#define cstring_find ( str,
                       pos,
                       ptr,
                       count,
                       ret_offset )
```

cstring_find - Find the first occurrence of the given substring.

Implements the Rabin-Karp algorithm.

Parameters

str - The cstring.

pos - Position at which to start the search, i.e. the found substring must not begin in a position preceding `pos`. Zero means that the whole `str` is searched.

ptr - Pointer to the first character of the string to search for.

count - Length of the string to search for.

ret_offset - Variable of type `ptrdiff_t` that receives the position of the first character of the found substring or -1 if no such substring is found.

Returns

void

◆ cstring_find_first_not_of

```
#define cstring_find_first_not_of ( str,
                                    pos,
                                    ptr,
                                    count,
                                    ret_offset )
```

cstring_find_first_not_of - Find the first character equal to none of the characters in the given character sequence.

Parameters

str	- The cstring.
pos	- Position at which to begin searching.
ptr	- Pointer to the first character of the string identifying characters to search for.
count	- Length of the string of characters to search for.
ret_offset	- Variable of type `ptrdiff_t` that receives the position of the first found character or -1 if no such character is found.

Returns

void

◆ cstring_find_first_of

```
#define cstring_find_first_of ( str,
                                pos,
                                ptr,
                                count,
                                ret_offset )
```

cstring_find_first_of - Find the first character equal to one of the characters in the given character sequence.

Parameters

str	- The cstring.
pos	- Position at which to begin searching.
ptr	- Pointer to the first character of the string identifying characters to search for.
count	- Length of the string of characters to search for.
ret_offset	- Variable of type `ptrdiff_t` that receives the position of the first found character or -1 if no such character is found.

Returns

void

◆ cstring_find_last_not_of

```
#define cstring_find_last_not_of ( str,
                                   pos,
                                   ptr,
                                   count,
                                   ret_offset )
```

cstring_find_last_not_of - Find the last character equal to none of the characters in the given character sequence.

Parameters

str	- The cstring.
pos	- Position at which to begin searching. -1 means that the whole `str` is searched.
ptr	- Pointer to the first character of the string identifying characters to search for.
count	- Length of the string of characters to search for.
ret_offset	- Variable of type `ptrdiff_t` that receives the position of the first found character or -1 if no such character is found.

Returns

void

◆ cstring_find_last_of

```
#define cstring_find_last_of ( str,
                               pos,
                               ptr,
                               count,
                               ret_offset )
```

cstring_find_last_of - Find the last character equal to one of the characters in the given character sequence.

Parameters

str	- The cstring.
pos	- Position at which to begin searching. -1 means that the whole `str` is searched.
ptr	- Pointer to the first character of the string identifying characters to search for.
count	- Length of the string of characters to search for.
ret_offset	- Variable of type `ptrdiff_t` that receives the position of the first found character or -1 if no such character is found.

Returns

void

◆ cstring_fix

```
#define cstring_fix ( str,
                      length,
                      value,
                      mode )
```

cstring_fix - Update the cstring to a fixed length by either padding or shortening.

Parameters

str - The cstring.

length - New length of the cstring.

value - Character used for the padding.

mode - Flags specifying where the cstring is to be padded or shortened.
1 at the begin of the cstring
2 at the end of the cstring
Their combination (1|2) leads to a centered alignment.

Returns

void

◆ cstring_free

```
#define cstring_free ( str )
```

cstring_free - Free all memory associated with the cstring and set it to NULL.

Parameters

str - The cstring. Can be a NULL string.

Returns

void

◆ cstring_front

#define cstring_front (str)

cstring_front - Return the first character in the cstring.

Parameters

str - The cstring.

Returns

The ASCII value (as `int`) of the first character in the string. If the macro fails, -1 is returned.

◆ cstring_init

#define cstring_init (name,
type)

cstring_init - Allocate a new cstring with zero length.

Also see `cstring_assign()`, `cstring_reserve()`, `cstring_push_back()`, `cstring_append()`, `cstring_resize()`.

Parameters

name - A not yet used variable name for the cstring variable to be declared and initialized.

type - The type of string to act on.

Returns

void

◆ cstring_insert

```
#define cstring_insert ( str,
                         pos,
                         ptr,
                         count )
```

cstring_insert - Insert a string at position `pos` into the cstring.

Parameters

str - The cstring.

pos - Position in the string where the new characters are inserted.

ptr - Pointer to the first character inserted into the cstring.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_iterator

```
#define cstring_iterator ( type )
```

cstring_iterator - The iterator type used for a string.

Parameters

type - The character type of the iterator to act on.

◆ cstring_length

```
#define cstring_length ( str )
```

cstring_length - Get the current length of the string.

Parameters

str - The cstring. Can be a NULL string.

Returns

The length as a `size_t`, terminating null not counted. Zero if `str` is NULL.

◆ cstring_literal

```
#define cstring_literal ( name,
                          type,
                          lit )
```

cstring_literal - Generate a cstring object with static duration.

> **Note**
> The pointer references static read-only data which is constant at compile time. DO NOT FREE IT.

Parameters

name - A not yet used variable name for the cstring object.

type - The type of string to act on.

lit - A string literal used to create the cstring literal. The argument passed to this parameter cannot be a pointer!

◆ cstring_max_size

```
#define cstring_max_size ( type )
```

cstring_max_size - Get the maximum number of elements a string of the specified character type is able to hold.

For clarity, this is like:
`((min(PTRDIFF_MAX, (SIZE_MAX / 2)) - sizeof(metadata)) / sizeof(type) - 1)`, with -1 for the string terminator that is not counted. Also, (SIZE_MAX / 2) because any `array + SIZE_MAX` would be bogus. PTRDIFF_MAX and SIZE_MAX may not be defined in ancient C libraries. Hence the calculation in the macro. However, the value of the macro is a constant expression. It is supposed to be calculated at compile time.

> **Note**
> The resulting value is technically possible. However, typically allocations of such a big size will fail.

Parameters

type - The type of string to act on.

Returns

The maximum number of elements the string is able to hold.

◆ cstring_pop_back

#define cstring_pop_back (str)

cstring_pop_back - Remove the last character from the cstring.

Parameters

str - The cstring.

Returns

void

◆ cstring_push_back

#define cstring_push_back (str,
value)

cstring_push_back - Add a character to the end of the string.

Also see `cstring_init()`, `cstring_assign()`, `cstring_reserve()`, `cstring_append()`, `cstring_resize()`.

Parameters

str - The cstring. Can be a NULL string.

value - The character to add.

Returns

void

◆ cstring_replace

```
#define cstring_replace ( str,
                          pos,
                          n,
                          ptr,
                          count )
```

cstring_replace - Replace a substring beginning at position pos with another string.

Parameters

str - The cstring.

pos - Offset of the first character replaced in the cstring.

n - Number of consecutive characters to be replaced.

ptr - Pointer to the first replacement character.

count - Number of consecutive replacement characters to be used.

Returns

void

◆ cstring_reserve

```
#define cstring_reserve ( str,
                          n )
```

cstring_reserve - Request that the string capacity be at least enough to contain n characters.

If n is greater than the current string capacity, the function causes the container to reallocate its storage increasing its capacity to n (or greater).
Also see `cstring_init()`, `cstring_assign()`, `cstring_push_back()`, `cstring_append()`, `cstring_resize()`.

Parameters

str - The cstring. Can be a NULL string.

n - Minimum capacity for the string.

Returns

void

◆ cstring_resize

#define cstring_resize (str,
count,
value)

cstring_resize - Resize the container to contain `count` characters.

Also see `cstring_init()`, `cstring_assign()`, `cstring_reserve()`, `cstring_push_back()`, `cstring_append()`.

Parameters

str - The cstring. Can be a NULL string.

count - New size of the cstring.

value - The value to initialize new characters with.

Returns

void

◆ cstring_reverse

#define cstring_reverse (str)

cstring_reverse - Reverse the character order in the cstring.

Parameters

str - The cstring.

Returns

void

◆ cstring_rfind

```
#define cstring_rfind ( str,
                        pos,
                        ptr,
                        count,
                        ret_offset )
```

cstring_rfind - Find the last occurrence of the given substring.

Implements the Rabin-Karp algorithm.

Parameters

str	- The cstring.
pos	- Position at which to start the search, proceeded from right to left. Hence the found substring cannot begin in a position following `pos`. -1 means that the whole `str` is searched.
ptr	- Pointer to the first character of the string to search for.
count	- Length of the string to search for.
ret_offset	- Variable of type `ptrdiff_t` that receives the position of the first character of the found substring or -1 if no such substring is found.

Returns

void

◆ cstring_shrink_to_fit

```
#define cstring_shrink_to_fit ( str )
```

cstring_shrink_to_fit - Request the container to reduce its capacity to fit its size.

Parameters

str - The cstring.

Returns

void

◆ cstring_size

#define cstring_size (str)

cstring_size - Get the current length of the string.

Parameters

str - The cstring. Can be a NULL string.

Returns

The length as a `size_t`, terminating null not counted. Zero if `str` is NULL.

◆ cstring_starts_with

#define cstring_starts_with (str,
ptr,
count,
ret_found)

cstring_starts_with - Check if the string begins with the given prefix.

Parameters

str - The cstring.

ptr - Pointer to the first character of the prefix.

count - Length of the prefix.

ret_found - Variable of type `int` that receives the result of the check.
0 if the prefix was not found
1 if the prefix was found

Returns

void

◆ cstring_string_type

#define cstring_string_type (type)

cstring_string_type - The string type used in this library.

Parameters

type - The character type of the string to act on.

◆ cstring_substring

```
#define cstring_substring ( from,
                            pos,
                            n,
                            to )
```

cstring_substring - Copy a part of a string.

Parameters

from - The source cstring.

pos - Position in the source cstring where the substring begins.

n - Number of consecutive characters copied to the substring.

to - Destination to which the substring is copied. Can be a NULL string.
If `to` refers to an existing cstring, the old content is overwritten.

Returns

void

◆ cstring_swap

```
#define cstring_swap ( str,
                       other )
```

cstring_swap - Exchange the content of the cstring by the content of another cstring of the same type.

Parameters

str - The original cstring. Can be a NULL string.

other - The other cstring to swap content with. Can be a NULL string.

Returns

void

◆ cstring_trim

```
#define cstring_trim ( str,
                       value,
                       mode )
```

cstring_trim - Remove contiguous occurrences of the specified character from the begin and/or the end of a cstring.

Parameters

str - The cstring.

value - The character to be removed.

mode - Flags specifying where the characters are removed.
1 to remove leading characters
2 to remove trailing characters
Their combination (1|2) results in trimming on both sides of the string.

Returns

void

◆ cstring_unsafe_set_size

```
#define cstring_unsafe_set_size ( str,
                                  size )
```

cstring_unsafe_set_size - Set the size property to the specified value and add the string terminator accordingly.

Providing a cstring with sufficiently large capacity as buffer to external API is supported. However, the third party API cannot update the header data of a cstring. This function enables the user to manually update the size property in order to keep it usable in the cstring API.

> **Note**
> This function does not examine the string data to evaluate the credibility of the specified size. Furthermore, this function does not force the capacity of the cstring to grow. If the required size exceeds the current capacity, the function sets the size to meet the capacity. Consider the memory being corrupted by the API that updated the string data in this case.

Parameters

str - The cstring.

size - The size to be applied.

Returns

void