The cstring_array API

Macros

#define **cstring_array_type**(type)
cstring_array_type - The type of the data in the vector of strings.

#define **cstring_array**(type)
cstring_array - Syntactic sugar to retrieve a vector type.

#define **cstring_array_iterator**(type)
cstring_array_iterator - The iterator type used for a vector.

#define **cstring_split**(str, max_tok, ptr, count, ret_array)
cstring_split - Tokenize a cstring into a cstring_array vector.

#define **cstring_array_free**(arr)
cstring_array_free - Recursively free all memory associated with the cstring_array and set it to NULL.

#define **cstring_array_at**(arr, pos)
cstring_array_at - Return the pointer to the first character of the string at position pos in the cstring_array.

#define **cstring_array_front**(arr)
cstring_array_front - Return the pointer to the first character of the first string in the cstring_array.

#define **cstring_array_back**(arr)
cstring_array_back - Return the pointer to the first character of the last string in the cstring_array.

#define **cstring_array_begin**(arr)
cstring_array_begin - Return an iterator to first string of the vector.

#define **cstring_array_end**(arr)
cstring_array_end - Return an iterator to one past the last string of the vector.

#define **cstring_array_empty**(arr)
cstring_array_empty - Return 1 if the vector is empty.

#define **cstring_array_size**(arr)
cstring_array_size - Get the current length of the vector.

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

#define **cstring_array_reserve**(arr, n)
cstring_array_reserve - Request that the vector capacity be at least enough to contain n strings.

#define **cstring_array_capacity**(arr)
cstring_array_capacity - Get the current capacity of the vector.

#define **cstring_array_shrink_to_fit**(arr)
cstring_array_shrink_to_fit - Request the container to reduce its capacity to fit its size.

#define **cstring_array_clear**(arr)
cstring_array_clear - Erase all of the strings in the vector.

#define	**cstring_array_insert**(arr, pos, ptr, count) cstring_array_insert - Insert a string at position `pos` into the vector.
#define	**cstring_array_erase**(arr, pos, n) cstring_array_erase - Remove the strings beginning at offset `pos` from the cstring_array.
#define	**cstring_array_push_back**(arr, ptr, count) cstring_array_push_back - Add a string to the end of the vector.
#define	**cstring_array_pop_back**(arr) cstring_array_pop_back - Remove the last string from the cstring_array.
#define	**cstring_array_copy**(from, to) cstring_array_copy - Copy a cstring_array.
#define	**cstring_array_resize**(arr, n, ptr, count) cstring_array_resize - Resize the container to contain `count` strings.
#define	**cstring_array_swap**(arr, other) cstring_array_swap - Exchange the content of the cstring_array by the content of another cstring_array of the same type.
#define	**cstring_array_slice**(from, pos, n, to) cstring_array_slice - Copy a part of a vector.
#define	**cstring_array_join**(arr, ptr, count, ret_str) cstring_array_join - Concatenate the strings of a vector using the specified joiner.

Detailed Description

Macro Definition Documentation

◆ cstring_array

#define cstring_array (type)

cstring_array - Syntactic sugar to retrieve a vector type.

Parameters

type - The character type of the strings in the vector.

◆ cstring_array_at

#define cstring_array_at (arr,
pos)

cstring_array_at - Return the pointer to the first character of the string at position pos in the cstring_array.

Parameters

arr - The cstring_array.

pos - Position of a string in the vector.

Returns

A string pointer at the specified position in the vector or NULL.

◆ cstring_array_back

#define cstring_array_back (arr)

cstring_array_back - Return the pointer to the first character of the last string in the cstring_array.

Unlike member cstring_array_begin, which returns an iterator just past this string, this function returns a direct string pointer.

Parameters

arr - The cstring_array.

Returns

A string pointer to the last string in the vector or NULL.

◆ cstring_array_begin

#define cstring_array_begin (arr)

cstring_array_begin - Return an iterator to first string of the vector.

Parameters

arr - The cstring_array.

Returns

An iterator to the first string (or NULL).

◆ cstring_array_capacity

#define cstring_array_capacity (arr)

cstring_array_capacity - Get the current capacity of the vector.

Parameters

arr - The cstring_array. Can be a NULL vector.

Returns

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

◆ cstring_array_clear

#define cstring_array_clear (arr)

cstring_array_clear - Erase all of the strings in the vector.

Parameters

arr - The cstring_array.

Returns

void

◆ cstring_array_copy

#define cstring_array_copy (from,
to)

cstring_array_copy - Copy a cstring_array.

Parameters

from - The original cstring_array.

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

Returns

void

◆ cstring_array_empty

#define cstring_array_empty (arr)

cstring_array_empty - Return 1 if the vector is empty.

Parameters

arr - The cstring_array. Can be a NULL vector.

Returns

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

◆ cstring_array_end

#define cstring_array_end (arr)

cstring_array_end - Return an iterator to one past the last string of the vector.

Parameters

arr - The cstring_array.

Returns

An iterator to one past the last string (or NULL).

◆ cstring_array_erase

#define cstring_array_erase (arr,
pos,
n)

cstring_array_erase - Remove the strings beginning at offset `pos` from the cstring_array.

Parameters

arr - The cstring_array.

pos - Offset of the first string erased from the cstring_array.

n - Number of consecutive strings to be erased.

Returns

void

◆ cstring_array_free

#define cstring_array_free (arr)

cstring_array_free - Recursively free all memory associated with the cstring_array and set it to NULL.

Parameters

arr - The cstring_array. Can be a NULL vector.

Returns

void

◆ cstring_array_front

#define cstring_array_front (arr)

cstring_array_front - Return the pointer to the first character of the first string in the cstring_array.

Unlike member cstring_array_begin, which returns an iterator to this same string, this function returns a direct string pointer.

Parameters

arr - The cstring_array.

Returns

A string pointer to the first string in the vector or NULL.

◆ cstring_array_insert

#define cstring_array_insert (arr,
pos,
ptr,
count)

cstring_array_insert - Insert a string at position pos into the vector.

Parameters

arr - The cstring_array.

pos - Position in the vector where the new string is inserted.

ptr - Pointer to the first character of the string inserted into the cstring_array.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_array_iterator

#define cstring_array_iterator (type)

cstring_array_iterator - The iterator type used for a vector.

Parameters

type - The character type of the strings in the vector.

◆ cstring_array_join

#define cstring_array_join (arr,
ptr,
count,
ret_str)

cstring_array_join - Concatenate the strings of a vector using the specified joiner.

Parameters

arr - The cstring_array.

ptr - Pointer to the first character of the string joining the elements of the cstring_array. Can be NULL.

count - Number of consecutive characters to be used. Can be zero.

ret_str - A cstring variable of the same character type as `arr` that receives the joined string. If `ret_str` refers to an existing cstring, the old content is overwritten.

Returns

void

◆ cstring_array_max_size

#define cstring_array_max_size (type)

cstring_array_max_size - Get the maximum number of elements a vector of strings of the specified character type is able to hold.

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

Parameters

type - The character type of strings in the vector to act on.

Returns

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

◆ cstring_array_pop_back

#define cstring_array_pop_back (arr)

cstring_array_pop_back - Remove the last string from the cstring_array.

Parameters

arr - The cstring_array.

Returns

void

◆ cstring_array_push_back

#define cstring_array_push_back (arr,
ptr,
count)

cstring_array_push_back - Add a string to the end of the vector.

Parameters

arr - The cstring_array. Can be a NULL vector.

ptr - Pointer to the first character of the string added to the cstring_array.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_array_reserve

```
#define cstring_array_reserve ( arr,
                                n )
```

cstring_array_reserve - Request that the vector capacity be at least enough to contain n strings.

If n is greater than the current vector capacity, the function causes the container to reallocate its storage increasing its capacity to n (or greater).

Parameters

arr - The cstring_array. Can be a NULL vector.

n - Minimum capacity for the vector.

Returns

void

◆ cstring_array_resize

```
#define cstring_array_resize ( arr,
                               n,
                               ptr,
                               count )
```

cstring_array_resize - Resize the container to contain `count` strings.

Parameters

arr - The cstring_array. Can be a NULL vector.

n - New size of the cstring_array.

ptr - Pointer to the first character of the strings added if the container grows.

count - Number of consecutive characters to be used.

Returns

void

◆ cstring_array_shrink_to_fit

#define cstring_array_shrink_to_fit (arr)

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

Parameters

arr - The cstring_array.

Returns

void

◆ cstring_array_size

#define cstring_array_size (arr)

cstring_array_size - Get the current length of the vector.

Parameters

arr - The cstring_array. Can be a NULL vector.

Returns

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

◆ cstring_array_slice

#define cstring_array_slice (from,
pos,
n,
to)

cstring_array_slice - Copy a part of a vector.

Parameters

from - The source vector.

pos - Position in the source vector where the part begins.

n - Number of consecutive strings to copy.

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

Returns

void

◆ cstring_array_swap

```
#define cstring_array_swap ( arr,
                             other )
```

cstring_array_swap - Exchange the content of the cstring_array by the content of another cstring_array of the same type.

Parameters

arr - The cstring_array. Can be a NULL vector.

other - The other cstring_array to swap content with. Can be a NULL vector.

Returns

void

◆ cstring_array_type

```
#define cstring_array_type ( type )
```

cstring_array_type - The type of the data in the vector of strings.

Parameters

type - The character type of the strings in the vector.

◆ cstring_split

```
#define cstring_split ( str,
                        max_tok,
                        ptr,
                        count,
                        ret_array )
```

cstring_split - Tokenize a cstring into a cstring_array vector.

Parameters

str - The cstring.

max_tok - Maximum number of tokens to be created. -1 specifies that all tokens are created.

ptr - Pointer to the first character of the delimiter string that separates the tokens in `str`.

count - Number of consecutive characters to be used as delimiter.

ret_array - Variable of **cstring_array(type)** that receives the created vector. Can be a NULL vector. If `ret_array` refers to an existing vector, the old content is overwritten.

Returns

void