Strings

This page contains documentation about string types, including the hashing functions native to blib.

String

```
typedef struct _MeasuredBuffer {
    DWORD length;
    char* buffer;
} String, MeasuredBuffer;
```

Initialisers

String* vallocString(PBYTE text)

> This function creates a deepcopy of `text` and does not free the original buffer.

Uses `valloc` to allocate `String`.

String* hallocString(PBYTE text)

> This function creates a deepcopy of `text` and does not free the original buffer.

Uses `halloc` to allocate a `String`.

Methods

unsigned int rapidStringscmp(String* source, Strings* samples)

Returns the index of the sample `String` within the `Strings` that matches the `String` 'source'. Else returns -1 if none is found.

unsigned int rapidncmp (String source, unsigned int n, String* samples)

Compares a `String` `source` against an array of `String`. Returns the index of the matching `String`, or -1 if none are found.

unsigned int rapidstrcmp (String s1, String s2)

Compares two `String`s, `s1` and `s2`. Returns `0` if both strings are the same, or the first index if they differ. Returns `-1` if the strings are not of the same length.

Generic `char*` Functions

Methods

DWORD bSplitToStrings(PBYTE string, char c, MeasuredBuffers** buffers)

> This function is unsafe and may result in accessing illegal memory if there are no gaurd null-bytes on the source.

> This function returns up to a maximum of BLIB_MAXIMUM_SUBSTRINGS defined at the blib compile time. Additionally, this returns a maximum size of BLIB_MAXIMUM_SUBSTRING_SIZE for each of the given substrings. This function does NOT include the nullbyte and performs an in-place copy, so a substring of BLIB_MAXIMUM_SUBSTRING_SIZE length will NOT have a guard null byte.

Returns the number of substrings when splitting the input 'string' on the character 'c'. This populates the 'buffers' pointer with an array of MeasuredBuffers. An example usage is provided below where the Strings struct is used as the `MeasuredBuffer**` array.

```
Strings* strings;
bWideCharToByte(cStr, index);
DWORD dwStringCount = bSplitToStrings(cStr, '=', &strings);
if( dwStringCount == 1 ){
    hfreeMeasuredBuffers(strings);
    index += length;
```

```
        continue;
    }
    cprintf("%s : %s\n", strings->members[0]->buffer, strings->members[1]->buffer );
```

DWORD bSafeWideCharToByte(PBYTE dest, const PWCHAR source, PDWORD buffSize)

> This function is unsafe and may result in accessing illegal memory if there are no gaurd null-bytes on the source.

If the destination is `NULL` this functions returns the buffer size required in 'buffSize'. Otherwise, this function performs the widechar -> byte conversion and returns the length of the new buffer uszed in buffSize. This function adds and calculates the null byte '\0' in the destination.

DWORD bWideCharToByte(PBYTE dest, const PWCHAR source)

> This function is unsafe and may result in accessing illegal memory if there are no gaurd null-bytes on the source.

Converts a source wide char to the designated destination.

unsigned int bstrlen(PBYTE string)

> This function is unsafe and may result in accessing illegal memory if there are no gaurd null-bytes.

Returns the index of the first nullbyte.

Revision #3
Created 2 August 2024 18:11:04 by lepus
Updated 11 August 2024 12:31:25 by lepus