```c
/**
* Stringbuilder - a library for working with C strings that can grow dynamically
  as they are appended
*
*/

#include <stdlib.h>
#include <string.h>
#include "stringbuilder.h"


/**
* Creates a new stringbuilder with the default chunk size
*
*/
stringbuilder* sb_new()
{
    return sb_new_with_size(1024);
}

/**
* Creates a new stringbuilder with initial size at least the given size
*/
stringbuilder* sb_new_with_size(int size)
{
    stringbuilder* sb;

    sb = (stringbuilder*)malloc(sizeof(stringbuilder));
    sb->size = size;
    sb->cstr = (char*)malloc(size);
    sb->pos = 0;
    sb->reallocs = 0;

    // Fill cstr with null to ensure it is always null terminated
    memset(sb->cstr, '\0', size);

    return sb;
}

void sb_reset(stringbuilder* sb)
{
    sb->pos = 0;
    memset(sb->cstr, '\0', sb->size);
}

/**
* Destroys the given stringbuilder
*/
void sb_destroy(stringbuilder* sb, int free_string)
{
    if (free_string)
        free(sb->cstr);

    free(sb);
}

/**
* Internal function to resize our string buffer's storage.
* \return 1 iff sb->cstr was successfully resized, otherwise 0
*/
int sb_resize(stringbuilder* sb, const int new_size)
{
    char* old_cstr = sb->cstr;

    sb->cstr = (char *)realloc(sb->cstr, new_size);

    if (sb->cstr == NULL)
    {
        sb->cstr = old_cstr;
        return 0;
    }

    memset(sb->cstr + sb->pos, '\0', new_size - sb->pos);
    sb->size = new_size;
    sb->reallocs++;
    return 1;
}

int sb_double_size(stringbuilder* sb)
{
    return sb_resize(sb, sb->size * 2);
}

void sb_append_ch(stringbuilder* sb, const char ch)
{
    int new_size;

    if (sb->pos == sb->size)
        sb_double_size(sb);

    sb->cstr[sb->pos++] = ch;
}

/**
* Appends at most length of the given src string to the string buffer
*/
void sb_append_strn(stringbuilder* sb, const char* src, int length)
{
    int chars_remaining;
    int chars_required;
    int new_size;

    // <buffer size> - <zero based index of next char to write> - <space for null ⤶
      terminator>
```

```c
    chars_remaining = sb->size - sb->pos - 1;
    if (chars_remaining < length)
    {
        chars_required = length - chars_remaining;
        new_size = sb->size;
        do {
            new_size = new_size * 2;
        } while (new_size < (sb->size + chars_required));

        sb_resize(sb, new_size);
    }

    memcpy(sb->cstr + sb->pos, src, length);
    sb->pos += length;
}

/**
* Appends the given src string to the string builder
*/
void sb_append_str(stringbuilder* sb, const char* src)
{
    sb_append_strn(sb, src, strlen(src));
}


/**
* Allocates and copies a new cstring based on the current stringbuilder contents
*/
char* sb_make_cstring(stringbuilder* sb)
{
    if (!sb->pos)
        return 0;

    char* out = (char*)malloc(sb->pos + 1);
    strcpy(out, sb_cstring(sb));

    return out;
}
```