Advanced Course Pre-Lesson

C String Library

Implement following functions by yourself with prefix “my_”, for example, my_memcpy, my_strcpy, my_strcat….

- **Copy**
  - **strcpy**

    *char * strcpy (char * destination, const char * source);*

    Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point).
  - **strncpy**

    *char * strncpy (char * destination, const char * source, size_t num);*

    Copies the first num characters of source to destination. If the end of the source C string (which is signaled by a null-character) is found before num characters have been copied, destination is padded with zeros until a total of num characters have been written to it.

    No null-character is implicitly appended at the end of destination if source is longer than num. Thus, in this case, destination shall not be considered a null terminated C string (reading it as such would overflow).

  Return Value:

  Destination is returned.
- **Concatenation**
  - **strcat**

    *char * strcat (char * destination, const char * source);*

    Appends a copy of the source string to the destination string. The terminating null character in destination is overwritten by the first character of source, and a null-character is included at the end of the new string formed by the concatenation of both in destination.
  - **strncat**

    *char * strncat (char * destination, const char * source, size_t num);*

    Appends the first num characters of source to destination, plus a terminating null-character.

    If the length of the C string in source is less than num, only the content up to the terminating null-character is copied.

  Return Value:

  Destination is returned.

- **Comparison**
  - **strcmp**

    *int strcmp (const char * str1, const char * str2);*

    Compares the C string str1 to the C string str2.
  - **strncmp**

    *int strncmp (const char * str1, const char * str2, size_t num);*

    Compares up to num characters of the C string str1 to those of the C string str2.

Return Value	***Indicates***
<0	The first character that does not match has a lower value in ptr1 than in ptr2.
0	The contents of both strings are equal.
>0	The first character that does not match has a greater value in ptr1 than in ptr2.

- **Searching**
  - **strchr**

    *const char * strchr (const char * str, int character); // C*

    *char * strchr (char * str, int character); // C++ improves*

    Returns a pointer to the first occurrence of character in the C string str.

    Return Value:

    A pointer to the first occurrence of character in str. If the character is not found, the function returns a null pointer.
  - **strrchr**

    *const char * strrchr (const char * str, int character); // C*

    *char * strrchr (char * str, int character); // C++ improves*

    Returns a pointer to the last occurrence of character in the C string str.

    Return Value:

    A pointer to the last occurrence of character in str. If the character is not found, the function returns a null pointer.
  - **strstr**

    *const char * strstr (const char * str1, const char * str2); // C*

    *char * strstr (char * str1, const char * str2); // C++ improves*

    Returns a pointer to the first occurrence of str2 in str1, or a null pointer if str2 is not part of str1.

    Return Value:

    A pointer to the first occurrence in str1 of the entire sequence of characters specified in str2, or a null pointer if the sequence is not present in str1.

- **strtok**

  *char * strtok (char * str, const char * delimiters);*

  A sequence of calls to this function split str into tokens, which are sequences of contiguous characters separated by any of the characters that are part of delimiters.

  On a first call, the function expects a C string as argument for str, whose first character is used as the starting location to scan for tokens. In subsequent calls, the function expects a null pointer and uses the position right after the end of the last token as the new starting location for scanning.

  To determine the beginning and the end of a token, the function first scans from the starting location for the first character not contained in delimiters (which becomes the beginning of the token). And then scans starting from this beginning of the token for the first character contained in delimiters, which becomes the end of the token. The scan also stops if the terminating null character is found.

  This end of the token is automatically replaced by a null-character, and the beginning of the token is returned by the function.

  Once the terminating null character of str is found in a call to strtok, all subsequent calls to this function (with a null pointer as the first argument) return a null pointer.

  The point where the last token was found is kept internally by the function to be used on the next call (particular library implementations are not required to avoid data races).

  Return Value:

  If a token is found, a pointer to the beginning of the token. Otherwise, a null pointer.

  A null pointer is always returned when the end of the string (i.e., a null character) is reached in the string being scanned.

- **Other**
  - **strlen**

    *size_t strlen (const char * str);*

    Returns the length of the C string str.

    Return Value:

    The length of string.

Reference: http://www.cplusplus.com/reference/cstring/?kw=string.h