

Re: Two Questions about "strlen", "strcat" and "strcpy"

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2004-08/3089.html

From: Paul Hsieh (*qed_at_pobox.com*)

Date: 08/30/04

Date: 30 Aug 2004 05:03:19 GMT

mattjack40@hotmail.com (Matt) wrote:

> *I have 2 questions:*

>

> 1. *strlen returns an unsigned (size_t) quantity. Why is an unsigned value more appropriate than a signed value? Why is unsigned value less appropriate?*

The reasoning is that unsigned values have a larger maximum value and that negative lengths don't have any meaning. So under the assumption that strlen can never fail and only needs to represent all possible outputs of a string length, size_t is the most appropriate output.

> 2. *Would there be any advantage in having strcat and strcpy return a pointer to the "end" of the destination string rather than returning a pointer to its beginning?*

As others have posted, incremental string concatenation is simplified by such a scheme. The claims of superior performance is kind of funny though -- while technically true, it misses the greater point that in fact calls to strlen(), implicit or not, is the real performance problem.

People may write strcpy() in "hand coded assembly language" if they want, but the semantics for it limits the advantage one can gain from doing this on most platforms (on modern x86 compilers you will gain nothing by doing this.) memcpy() on the other hand, can be *drammatically* improved in performance using assembly language on most platforms -- the reason is that aligned block copying is something most hardware has good support for that is far superior to char by char copying.

So returning an "end pointer" helps half of the problem by implicitly tracking the end-address for the destination, but does nothing about the other half of the problem of not knowing the length of the source and thus still doing the *implicit strlen()* as part of the strcat or strcpy. If, on the other hand, the length of your source and

comp.lang.c: Re: Two Questions about "strlen", "strcat" and "strcpy"

destination strings is known beforehand, then one could use memcpy() instead, which would lead to a *true* performance boost.

People who have seen me post here before already know the punchline. I've written a string library that does precisely this sort of thing (as well as all sorts of other things related to speed, safety, functionality and maintainability). You can learn more by visiting the second link below.

--

Paul Hsieh

<http://www.pobox.com/~qed/>

<http://bstring.sf.net/>

--

comp.lang.c.moderated - moderation address: clcm@plethora.net