

Re: how to replace a substring in a string using C?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-11/msg00738.html

- *From:* Netocrat <netocrat@xxxxxxxxxxx>
 - *Date:* Wed, 09 Nov 2005 04:21:35 GMT
-

On Tue, 08 Nov 2005 05:20:04 -0800, Mark F. Haigh wrote:

> Netocrat wrote:

[...]

> There are a couple of problems here. First, [i should be size_t not int].

Well spotted – no one else caught that one (I only picked it up after Dave Thompson's critique).

> Second, this code performs poorly even though it may look at a glance to
> be somewhat optimized.

Yes; I've since posted code that's an improvement – it doesn't use
subscripting in the replacement loop and it uses memcmp rather than
strstr – but I realise that both you and Skarmander are correct that
character-by-character operations should be avoided. As Skarmander points
out my attempt to avoid redundant iteration did not in fact do that since
memcmp is called on every character.

[...]

> To give an idea of the impact of string parsing missteps can make on
> program performance, I fed "Much Ado About Nothing" (the official play
> of comp.lang.c) through your replace function, then through mine. The
> entire file (121K) was loaded into a buffer, then fed 10 times to the
> replace function as such:

>

> /* ... */

> fread(str, 1, size, fp);

> str[size] = '\0';

> for(i = 0; i < 10; i++) {

> newstr = replace(str, "DON PEDRO", "NETOCRAT"); free(newstr);

> }

> /* ... */

I've written a benchmark program, which I've made available here:

<http://members.dodo.com.au/~netocrat/c/source/replacebench.c>

It incorporates my fixed-up function, your function below and Skarmander's
suggested snippet (which has an identical algorithm to yours although the
expression is slightly different, and it uses memcpy where you've used

Re: how to replace a substring in a string using C?

This can be avoided (I know you've already qualified this as a somewhat naive implementation).

```
> size_t len_old = strlen(old);
> size_t len_new = strlen(new);
> size_t count;
>
```

If `len_old == len_new`, there's no need to run this loop and you can simply `malloc len_str + 1` bytes.

```
> for(count = 0, p = str; (p = strstr(p, old)); p += len_old)
> count++;
>
> ret = malloc(count * (len_new - len_old) + len_str + 1);
> if(!ret)
> return NULL;
>
> for(r = ret, p = str; (q = strstr(p, old)); p = q + len_old) {
> count = q - p;
```

As discussed elsewhere, under C99 this invokes undefined behaviour when `q - p > PTRDIFF_MAX`. I vaguely recall discussions on the semantics under C89, but can't recall the conclusion. Also under C99 `count` would be better declared `ptrdiff_t` but it appears that you're writing to C89.

```
> memcpy(r, p, count);
> r += count;
> strcpy(r, new);
```

As Dave T pointed out `memcpy` is more appropriate here; but as compiled under `gcc` on my platform `strcpy`'s very slightly faster than using `memcpy` – go figure.

```
> r += len_new;
> }
> strcpy(r, p);
> return ret;
> }
>
```

```
> Here are the timings on my machine. Both were compiled with the same
> flags on gcc 4 (-Wall -O2 -ansi -pedantic)
[results omitted]
> Even though the somewhat naive program traverses the input string
> multiple times, it's still a 99.9% execution time decrease.
```

Here's the optimised C99 version of the function used for the benchmark results above. The `#error` seems to me to be the best way of dealing with the possibility of UB – it alerts the programmer to it at compile time and encourages him/her to examine the comments in the code whilst removing it.

Re: how to replace a substring in a string using C?

```
#include <string.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdint.h>

# if PTRDIFF_MAX < SIZE_MAX
/* If the following line causes compilation to fail,
 * comment it out after taking note of its message.
 */
# error The replace function might invoke undefined \
behaviour for strings with length greater than \
PTRDIFF_MAX – see comments in the function body
# endif

char *replace(const char *str, const char *old, const char *new)
{
    char *ret, *r;
    const char *p, *q;
    size_t oldlen = strlen(old);
    size_t count, retlen, newlen = strlen(new);

    if (oldlen != newlen) {
        for (count = 0, p = str; (q = strstr(p, old)) != NULL; p = q + oldlen)
            count++;
        /* this is undefined if p - str > PTRDIFF_MAX */
        retlen = p - str + strlen(p) + count * (newlen - oldlen);
    } else
        retlen = strlen(str);

    ret = malloc(retlen + 1);

    for (r = ret, p = str; (q = strstr(p, old)) != NULL; p = q + oldlen) {
        /* this is undefined if q - p > PTRDIFF_MAX */
        ptrdiff_t l = q - p;
        memcpy(r, p, l);
        r += l;
        memcpy(r, new, newlen);
        r += newlen;
    }
    strcpy(r, p);

    return ret;
}
```

—
<http://members.dodo.com.au/~netocrat>

Re: how to replace a substring in a string using C?

- *Follow-Ups:*

- ◆ [Re: how to replace a substring in a string using C?](#)

- ◇ *From:* pete

- *References:*

- ◆ [Re: how to replace a substring in a string using C?](#)

- ◇ *From:* Mark F. Haigh

- Prev by Date: [Re: Why C?](#)

- Next by Date: [Re: Why C?](#)

- Previous by thread: [Re: how to replace a substring in a string using C?](#)

- Next by thread: [Re: how to replace a substring in a string using C?](#)

- Index(es):

- ◆ [Date](#)

- ◆ [Thread](#)