

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

## Re: Results of the memswap() smackdown from the thread "Sorting" assignment

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.programming/2008-02/msg00480.html>

---

- *From:* Randy Howard <[randyhoward@xxxxxxxxxxxxxxxxxxxx](mailto:randyhoward@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 12 Feb 2008 21:23:43 GMT
- 

On Tue, 12 Feb 2008 06:41:21 -0600, spinoza1111 wrote  
(in article  
<[def77da1-da1a-4713-9106-3636e78f8df4@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:def77da1-da1a-4713-9106-3636e78f8df4@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>):

On Feb 12, 6:39 pm, Randy Howard <[randyhow...@xxxxxxxxxxxxxxxxxxxx](mailto:randyhow...@xxxxxxxxxxxxxxxxxxxx)>  
wrote:

Richard's use of a "buffer" is clearly also a win for this benchmark (but not as good as Ben's optimization) but I remain opposed to it for the reasons I have given already.

That's pretty amazing to see, given that Ben's "version" is basically the one you're complaining about with an optimization up front for small swaps.

The problem was that C forces the use of an assembler style, with the threat of a return to GO TO, to implement Ben's optimization.

goto in C, any "GO TO" feature of a language isn't a threat. A jmp instruction is a goto. Look at the assembler output of your program sometime. It's filled with gotos. Language syntax just hides it for you. Goto isn't /always/ bad.

Yes, almost everyone has read Dijkstra's 1968 paper. He talks about the "unbridled use of the go to statement", and that is a valid concern. A lot of people have thought about it a lot more in the last 40 years and have managed to convince themselves that it isn't an outright ban as some like to pretend. The people claiming such quite often haven't even read it, they just heard it somewhere, often from an early CS professor.

For some more extensive discussions, consider these:

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

<http://www.ecn.purdue.edu/ParaMount/papers/rubin87goto.pdf>

(The part by Frank Rubin, starting halfway down the third column)

To which of course Dijkstra wrote the following response:

<http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1009.html>

David Tribble wrote a more recent take on it, in rather long form:

<http://david.tribble.com/text/goto.html>

There are plenty more of course, but its safe to say that the jury is still out on it being a "ban". Finding offense because a language construct like switch() can be used in a fashion that reminds you too much of a goto is a weak attempt.

And by the way, it's not an "assembler style". You want to see the real deal? Take a look at memcpy.s on a modern OS with an open source library. Both of the versions your groaning about are heaps more readable, not to mention of course being portable. To get anything like that performance, you have to either suffer inline assembler, or stop being upset at the notion of calling memcpy() from the lib.

Did you really not notice?

It is in a contradictory fashion  
relying on the quality of whatever library does memcpy  
while at the  
same time it extends the library;

Hmm, apparently you really didn't. Look at them both again. How does the one calling memcpy() that you do like not suffer as the one calling memcpy() that you don't like? That's some interesting bias you have showing there.

Ben should have ended with Malcolm's or my own code and not used the Heathfield stunt. So you're right.

Well, at least you admitted it. Isn't amazing how much those bias blinders can obscure reality and inhibit rational thought?

if memcpy starts to suck over the  
lifetime of the program (which of course can include changes  
to the  
library) this will suck,

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Yeah, it's so incredibly common in practice for memcpy() to be deoptimized by updates and nobody bothers to notice or fix it. No wait a minute, it's not common at all.

I've seen it, and the real problem is that it can be done. What part of "risk and exposure costs through data systems that we don't know about, yet" don't you understand?

I understand the issues quite well. You must learn to balance risk, and the risk of that happening is imo considerably less than the risk of a home-grown implementation being flawed for something that is typically heavily optimized on every platform.

meaning that the "library" considered as now including the Heathfield memswap() will show perverse behavior over time, adding to the cost of software maintenance. My theorem being that for the same reason each library function should be a black box, its performance should be either independent of or smoothly and predictably related to other library functions, so that change of platforms don't generate research and rework.

If you follow through on this, you wind up reinventing all of the wheels in the libc, so as not to be "dependent" upon them for this goal

Actually, that won't happen, mostly because I won't do it. I would say that a more optimized library won't do a lot of internal reuse.

So you don't even follow your own advice. Well, I guess that should have been predicted. Doom on me.

Trading all of that for some mythical concern over the remote chance of a future version of the library suddenly getting much worse and nobody noticing it, seems completely misdirected, imo.

It remains a mistake to confuse cleverness with a better algorithm.

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Clearly. Your XOR attempt was the proof of that. We do have relatively simple solutions, that do not involve tricks, merely use of an existing age old API everyone knows, and they work better to boot.

Fortunately, it's not difficult to combine small functions in the standard library to perform more complex actions. In fact, that's the whole point.

It is the whole point, but if any one C program can change the rules, it's a waste of time.

Change what rules? If you think the ability of a program to do something not already provided on the platform is a downside, then you would logically conclude that nobody should write any software at all. I can sort of see why you might arrive at such a conclusion, but many others, including myself don't see it that way. ;-)

C in my experience makes fools out of those who would create tools

It does seem to do a good job of making the fools obvious, I'll grant you that one.

I don't know who you are talking about. Certainly, if a language moronizes people, it's the language's fault,

If someone goes jogging with a hunting knife hanging on a rope loop around their neck, and one day they trip and fall, and the knife stabs them through the heart, is it the knife at fault, or did they simply employ it in a moronic fashion?

People misuse tools every single day, a fact which keeps the legal profession extremely busy, arguing that their morons should be compensated because an inanimate object couldn't protect said moron from their own stupidity.

Programming languages are no better than any other tool at reversing such problems.

likewise if it seems to make fools wise.

A mere look at the warning label on a newly purchased clothes iron,

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

that warns you not to attempt to iron an item of clothing /while you are wearing it/ should lead you in the right direction. :)

Here, knowing C's mistakes is ersatz for insight, and posters other than myself have complained bitterly about this.

There is nothing confusing about the functions under discussion. They are dreadfully simple really. Ben's performance optimization for small swaps isn't completely novel even, such things have been done in C and many other languages for ages.

comp.programming should be C-free.

First, practice what you preach. Second, I see no reason for you to decide what languages can appear in c.p, especially given that C is one of the most widely used languages on the planet, and has been for decades.

You might as well ask a group discussing automobiles to refuse to talk about Fords, because the Pinto had a problem with exploding.

it overexposes things globally,

No, it doesn't.

Yes (sigh) it does. What part of #define don't you understand? What part of "scope"?

Then you really don't understand "global" at all.

```
-- xyz.c --  
  
#define FOO 27  
  
/* some more code */  
....  
  
-- end of xyz.c --  
  
  
-- main.c --  
  
#include <stdio.h>
```

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

```
int main(void)
{
printf("%d\n", FOO);
return 0;
}
```

— end of main.c —

Hmm, why can't these be compiled and linked, clearly in your universe, #define is a global. What part of /scope/ do you not understand?

More importantly, why do you not realize that macro definitions are optional, and if you want a narrower scope for something, you don't have to use them. C doesn't /force/ people to use it. You could easily write the functions under discussion without using them at all.

However, a primary usage of #define macros is to have a convenient way to eliminate magic numbers from code, so that the same number (64 in this case) doesn't get splashed all over the place, which is a maintenance concern.

you need to be far more expert in C details than I ever became to be a C toolsmith.

I agree.

What do you want me to say? That you're a greater man that I am? Fat chance at this point, buddy.

I expected nothing of the sort. I simply agreed with you. Sadly, and as we have seen before, you sometimes get angry when people agree with you. If this is a transference of anger over you not getting an argument, or not, I can't tell from here.

This is not based on personalities apart from Richard's modus operandi in coding, which is, at times, and in my opinion, to be clever in a fashion that for me is no longer useful

By your own admission, Ben's improvement on Richard's code was even more clever, but for some reason you like it anyway, and even forgive it for the very same traits that you find fault with in Richard's

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

original. Even a complete noob programmer could recognize the bias there.

I am biased, if that's the word, against Richard's lack of general culture, his superficial corporate-level view of the world, his religious enthusiasms, his confusion of clever code with a better algorithm, and his continuing efforts to control this newsgroup.

Unfortunately, /none/ of that has (or should have) any impact on an algorithms discussion. You keep claiming that you want to raise the level of discourse, to stop all of the mudslinging as it were, yet every time you have an opportunity to do so, you take a pass. Hypocritical, at best, outright lying at a worst interpretation.

I'd only add that "many operating systems" freeze up and piss me off when I am trying to get something done.

Get a better one. I'm extremely pleased with the one I'm using atm, but I have seen some real bad ones. Most came from the same vendor in the northwestern part of the US. Think that's coincidence, or perhaps climate related? ;-)

The Mac freezes and goes boing too.

It's possible. I've been running this one for just shy of 4 years now, and it hasn't happened to me yet. /\* crosses fingers \*/

I can't recall such stability on any modern OS platform, in fact, I'd have to go back to a version of AT&T SVSV R4 to find similar long-term stability, and that, while being a great platform, had nowhere near the feature set of this one, so I am, again, quite happy with it with what I have at the moment.

I've of course ran about 20-30 different versions/distros of Linux, and while many of them have also been more stable than that most popular one we all know, none have been as stable and as convenient to use as this one. I still have systems running Windows, Linux, OS X and some others at the moment, primarily for portability reasons and testing. I also hold out hope that one of them will emerge as even better than my current platform of choice. YMMV.

Anyway, if I had to extend the C library, I wouldn't do a lot of "reuse" solely for the purpose of speed as RH has done, at

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

least not  
as my first try.

You miss the point. I don't think he even mentioned speed originally.  
That was your addition, iirc.

You sleaze!

Are you setting an example of being collegial again?

He had to be otherwise he had no case. He was saying, in a patronizing way and to McLean, that C could do multibyte memswaps.

It wasn't patronizing at all. It simply was an implementation of a larger than byte-by-byte memswap, (provided that bufsize macro was >1). It was a response to a claim, that demonstrated the claim was incorrect. As I recall, and feel free to correct me if i'm wrong by showing a direct quote, he didn't mention performance at all. Which is exactly what I wrote above. And you call /me/ a sleaze?

But  
unless his code is faster than a simple for loop, his claim had no value or truth content, because C can do multibyte memswaps only by virtue of Turing's result, that any computer can do anything computable in some amount of time and with some amount of memory.

But? You benchmarked it yourself. You already know.

I am getting real tired here of unscientific claims being made selectively to wage campaigns of personal destruction.

Then stop making them.

Richard's algorithm was not an unscientific claim. It was an implementation of a solution to a problem that someone else said wasn't doable in the language.

Clive got away with using strlen in a for clause without so much as a peep from you thugs whereas when I did so in 2003,

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

He did? If so, I missed it. Could you point me to a link to the original article containing his code, and not your characterization of his code?

you were libeling me to Apress,

Yet another false, unscientific claim. Your Hypocrisy Disorder is acting up again.

because you respect him,

I've never met him, but he seems to be based upon what little knowledge I have of him, the bulk of which was obtained in the last few weeks in this newsgroup, a rather reasonable sort, and certainly has made some valuable contributions to the discussions. Especially the historical information about other languages and motivations inside the std committee.

because he tried to destroy Schildt.

I had read the review before, certainly. I didn't know who the author was, or what, if any credentials he had at the time I read it. But, I did find that I agreed with his review, since I had purchased the book in question, and found it completely unsatisfying, well, at least every other page of it.

At that time, I afforded him the same "respect" I would for anyone that wrote a book review and who I was not acquainted with.

His more recent articles here have actually raised my opinion of him, fwiw, because I now know more about him than I did at that time, but I still don't know him in any real sense.

all that seems to matter to you is Richard's ability to confuse the separate subjects of computer science and algorithms versus clever ape coding in C by making a false claim about C.

You really misunderstand what matters to me then. Would you like to know that I think /none/ of the proposed solutions should be used /as is/ in a production application, including the one by Richard, that you think I fawn over so much? These are minor nits, and do nothing but get in the way of your drag race challenge, but they do exist.

## Re: Results of the memswap() smackdown from the thread "Sorting" assignment

It is intellectual dishonesty of the first order to say that a computer or programming language is the only one you'll ever need because it can do anything. It's a salesman's claim.

I agree. Do you know of anyone that makes such a claim? If they are around, why don't the post here so we can all laugh at them?

The question was, not whether C can memswap (by way of Turing we know it can). It was "what is the best algorithm in abstract C, factoring out the many different library implementations, of swapping memory?"

That question wasn't asked at all, by anyone, until right there. I see you attempting to pick up the goal post and drag it downfield, but apparently it's too heavy, because the goalpost is having none of it.

The answer is from Nilges, Bacarisse and McLean and not Heathfield.

Once again, you are apparently irrevocably confused. Bacarisse's solution is almost identical to Richard's in the context of the "new question" you propose above. From this we can only conclude that your memory is horribly short, because you have already admitted that, just a few minutes earlier. I don't know how to deal with such delusions, maybe someone else can break through them.

It may in fact be the case to be bad practice to write a C library which uses any other library function. This is because you could never ensure that the reused function would not be redefined somewhere in the library source. My understanding is that conceptually, the source of the library is presented every time any program is compiled in such a way that I can redefine any library function by a macro.

Go ahead and try to write a real one that does anything worthwhile sometime while making no standard library calls. Then get back to us on your theory and your understanding.

Therefore it is professional malpractice to make such overblown claims as RH frequently makes for C.

What overblown claim did he make? Did his solution not meet the original spec? Did his solution not get the correct answer? Did his

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

solution not run quite well on test data?

Contrast that with the laughable and broken one you attempted. Contrast the last version of your code with his, and look at your own benchmark results (which you asked for, btw). Which was closer to "malpractice"?

It is personal discourtesy to base campaigns of personal destruction on these false claims. It is vandalism to dominate comp.programming with C programming.

I keep forgetting. Talking about C programming is off topic. Talking about psychology and philosophy and corporate abuse and politics is on topic, and not vandalism. Right. It seems you're not finding a lot of agreement though. You are however well within your rights to propose the addition of a new newsgroup, and you could even make it a moderated one, where you are in charge of the discussion completely.

In fact, I encourage you to do so. No doubt you would be happier, and all of your thousands of followers would probably be happier too. ;-)

RH clearly meant to show that C could do it fast IF you used a buffer.

No, he clearly meant to show that it was possible to exceed a byte-by-byte swap in standard C. That it would be faster is a likely outcome, for what /should/ be obvious reasons, and that was also the basis for Malcolm commenting upon it in the original, if you bothered to actually think about that. Anyone with any degree of experience with the field would immediately /conclude/ that it would likely be faster, but would of course perform experiments to find out. These experiments would be much more exhaustive than those applied so far here if portability across operating systems, compilers and architectures was important.

This was an admission that C in fact provides no fast path without jiggery-pokery

Use existing, standardized function calls from the language's own library is not "jiggery-pokery" to anyone even remotely familiar with the language. It's expected. Otherwise, such interfaces wouldn't even /be/ in the standard. What point is there in providing APIs if you aren't supposed to use them? Delusional.

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

because the string encapsulation in the latter means that specific implementations have a clear way, one not dependent on the competence of applications programmers, of optimizing large string moves.

What about memswap() implies it is only valid or valid /at all/ for general strings? More to the point, how can it even be valid to "swap strings" at all, unless you know ahead of time that they are equal in size? Do you have any idea how ridiculous this little bit sounds?

Hint: Look at your own implementation. Do you see void\*, or char \* in the interface?

You're eight track just skipped. Cycle back around to memswap() when you feel you can.

An object oriented memswap with state could remember, with complete safety, previous swap jobs such that the string lengths are suitably bounded using RH's trick with far less risk. This code would operate not as redefined potentially by some clown but according to the Java or C sharp contract.

You are obviously confused about the purpose of this memswap() function, so I'll give you time to go back and think about it, rather than point out more of the obvious.

He could have written his own memcpy() implementation to avoid using the libc function, but odds are in a cross-platform test, it would lose out to many, if not all of them since they would be tuned for the target architecture in ways often not possible with pure portable code that doesn't leverage the standard library implementation.

This certainly shouldn't be news today. Most knew it ages ago.

Here are the CPU rankings

Ben's code took 63 clock() values to do the exchanges 10000 times.

spinoza1111's code took 203 clock() values to do 10000 exchanges

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Richard's code took 109 values. to do 10000 exchanges

Malcom's code took 187 clock values. to do 10000 exchanges

A trend is emerging in these drag races you keep encouraging. Do you see it?

I'm not encouraging NASCAR meets. That would be you. You are continually viewing too many issues through the lens of dominance and control.

You're the one that keeps having contests, largely as a result of making claims that subsequently turn out to be false, in all cases btw, after empirical analysis. You might expect someone to get tired of smashing their face into the brick wall of reality all the time, but for some reason you relish in it. Trying to pretend that your approach is better than Heathfields, when your approach was broken, isn't going to wash, so I'm going to just skip all that...

I wanted Visual Basic, before and after .Net, to process strings in the way Rexx processes strings. In particular, I wanted to be able to pick out individual words in strings independent of the number of extra spaces between words. So, I wrote a tool to do so.

Again, strings and memswap() don't really belong together.

Game, set and match.

Yep, you proved that you don't understand this stuff. I won't argue with that.

```
#include <time.h>
#define ITERATIONS 100000
```

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

```
int main(int argc, char *argv[])
{
    printArray();
    int intIndex1;
    printf("%d\n", clock());
}
```

What type does clock() return?

Yes, we need to take your suggestion and divide it by clocks per second. That was left to the reader.

Oh by the way, it's broken to assume that clock's return value is compatible with %d. Don't let that get in your way though, that's just jiggery-pokery stuff.

My solution, and Malcolm's, are the best because they buy speed WITHOUT any dependencies on a legacy library and programming language that cause more trouble than they are worth.

Your solution doesn't seem to "buy speed" at all. If it does, it isn't getting a very good bang for the buck.

Ben's will be the best once he figures out a way of not falling through a case statement, since this is bad style from the point of view of the casual program reader who, in this day and age, may not understand the details of a poorly designed construct.

Programs source code is not intended for casual readers, philosophers or armchair psychologists. It is intended for programmers.

You might note, if you took your blinders off for a second, that you're the only one that thought the code was syntactically invalid. Presumably because of another example of you "forgetting more about C than anyone else has ever known"??

--

Randy Howard (2reply remove FOOBAR)  
"The power of accurate observation is called cynicism by those who have not got it." – George Bernard Shaw

Re: Results of the memswap() smackdown from the thread "Sorting" assignment

Re: Results of the memswap() smackdown from the thread "Sorting" assignment