

Re: The dangers of replacing malloc()

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-06/0117.html>

From: Terry Lambert (tlambert2_at_mindspring.com)

Date: 06/27/03

Date: Fri, 27 Jun 2003 03:23:38 -0700

To: "D. J. Bernstein" <djb@cr.yp.to>

"D. J. Bernstein" wrote:

- > *You obviously aren't claiming that POSIX requires `_all_` system functions*
- > *to use a replacement `malloc()/realloc()/free()` library for allocation.*
- > *That would prohibit `sbrk()` itself, for example.*

No. I'm claiming that it requires all `_POSIX_` functions to do so. The function `sbrk()` (actually, `brk()` is the function of interest, on FreeBSD) is an implementation detail on systems that use it.

- > *Are you trying to say that POSIX requires all `_POSIX functions_` to do*
- > *their memory allocation via `malloc()/realloc()/free()`? This has no*
- > *relevance to anything I said. We aren't talking about programs that*
- > *restrict themselves to the POSIX functions.*

Then we are talking about programs that have, and will continue to have, the problems that you are complaining about.

- > > *Suppose the OS distributor decides that `valloc()` or `xyzalloc()` should do*
- > > *its own thing, rather than calling `malloc()`.*
- > > *Then that OS distributor's OS no longer complies with standards.*
- >
- > *I already showed you code demonstrating that the Linux `valloc()` works*
- > *this way.*

You showed me code that valled `valloc()`; you didn't show me code *implementing `valloc()` on Linux*. I chose not to make this point because it would may have been a strawman, and I didn't need it to support my argument.

- > *If, as you claim, there's a ``standard'' prohibiting the Linux*
- > *behavior, then that ``standard'' is not useful for people who care about*
- > *real-world portability. Anyway, I see no evidence supporting your claim.*

I think you are thinking of SunOS, which does have `valloc()`, and not Linux (neither SuSE nore RedHat have it, according to their manual pages). FreeBSD doesn't have it, and POSIX doesn't specify

it.

My argument in this case is that the valloc() interface is not portable, and you should not use it. If you are not interested in portability, then you argument about the non-portability of carrying around your own library functions for memory management fails. Either way, your argument fails.

> > *If you are in non-compliance with the Intel Application Binary Interface
> > specification, you should expect to *minimally* be required to relink,
> > recompile, or have to modify your program source code, each time the
> > OS major version number changes*
>
> *If you link statically, upgrades can break your _source code_?*
>
> *Wow. I never realized that the -static option had such power. Will it
> also cause hair to grow on your palms?*

That's because you never tried to run a SunOS 4.4u2 binary which used the select(2) system call on SunOS 5.0, which returns ENOSYS if you attempt to call that entry point. If you had, you'd realize that upgrades can break you compiled code, and that you should therefore expect to be required to relink, recompile, or have to modify your program source code.

I'll note here that it's strange that you are complaining about a library using the brk(2) system call behind your back, but at the same time you appear unconcerned about libraries using the select(2) system call behind your back, both of which can result in identical classes of breakage.

I understand why you want what you want, but I'm going to tell you here and now that you are not going to find it in a general purpose OS unless you are willing to carry around your own code for things which may be implemented differently at the OS vendors discretion, while not preventing them from complying with standards.

As a final note on portability, I'll state the obvious: the portability of any program is inversely proportion to the number of system interfaces and system interface behaviours upon which it depends.

I personally worked on the first shrink-wrap product ever sold by a third party vendor for UNIX systems, and it was, during the heyday of the UNIX incompatibility wars, ported to over 140 different UNIX variants. Its hard system interface requirements were limited to 6 encapsulation functions, for which there were a grand total of three implementations, which fanned out to about 18, which breaks down to not using about 85% of the available system calls.

freebsd-performance: Re: The dangers of replacing malloc()

Most ports of this code took about 4 hours, and 3 of those hours were running a validation suite and reading the tape with the source code and writing several copies of the tape with the binary distribution and dirty build tree.

PS: The software didn't use valloc(), but it did use malloc()/free().

-- Terry

freebsd-performance@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-performance>

To unsubscribe, send any mail to "freebsd-performance-unsubscribe@freebsd.org"