

The dangers of replacing malloc()

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-06/0099.html>

From: D. J. Bernstein (*djb_at_cr.yo.to*)

Date: 06/26/03

Date: 26 Jun 2003 03:41:38 -0000

To: freebsd-performance@freebsd.org

Suppose that, taking the advice of inexperienced programmers who trumpet weak linking, I use `sbrk()` to write my own `malloc()`, `free()`, etc. Here's what can go horribly wrong.

Suppose the OS distributor doesn't go to the effort of dealing with competition for `sbrk()`. This is the normal situation; it has been quite explicitly tolerated by the `sbrk()` documentation for several centuries.

Suppose the OS distributor decides to write `somedorkyosfunction()` using some funky new allocation function that I haven't replaced because I haven't heard of it. Yesterday it was `valloc()`; tomorrow `xyzalloc()`. This happens all the time: look at the FreeBSD `realloc()`, for example.

Suppose the OS distributor decides that `valloc()` or `xyzalloc()` should do its own thing, rather than calling `malloc()`. This happens too: I tried the sample program shown below under Linux, and `somedorkyosfunction()` ended up calling `brk()` rather than my own `malloc()`.

Finally, suppose the OS distributor decides that some syscall I use should be replaced by a library routine that uses `somedorkyosfunction()`. This happens too. Note for the reading-impaired: I'm not saying that the name `malloc()` has ever been used for a syscall; I'm saying that `poll()`, `socket()`, et al. have been used for allocating library routines.

Result: My program innocently calls that library routine, which calls `somedorkyosfunction()`, which calls `valloc()` or `xyzalloc()`, which incorrectly assumes that its `sbrk()` results are contiguous, destroying the data allocated by my own `malloc()`.

As I said before, I encounter more than enough portability problems without going out of my way to look for them. I wish OS distributors would put a little more thought into the needs of people who `_don't_` spend their entire lives working with a single platform.

—D. J. Bernstein, Associate Professor, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago

freebsd-performance: The dangers of replacing malloc()

```
#include <stdlib.h>

void somedorkyosfunction(void)
{
    valloc(1);
}

void *malloc(size_t n)
{ write(1,"malloc\n",7); return 0; }
void *calloc(size_t n,size_t m)
{ write(1,"calloc\n",7); return 0; }
void *realloc(void *x,size_t n)
{ write(1,"realloc\n",8); return 0; }
void free(void *x)
{ write(1,"free\n",5); }

int main()
{
    malloc(1);
    realloc(0,1);
    somedorkyosfunction();
    return 0;
}
```

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"