

RE: ten thousand small processes

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/performance/2003-06/0075.html>

From: Michael E. Conlen (*meconlen_at_obfuscated.net*)

Date: 06/22/03

To: "D. J. Bernstein" <djb@cr.yp.to>, freebsd-performance@freebsd.org
Date: Sun, 22 Jun 2003 03:50:32 -0400

If your going to get this serious about your memory management, couldn't you just brk yourself and manage it your self? You seem to know exactly what your looking for and expect a specific result. I wouldn't recommend it to most, but you seem to know what your doing.

--

Michael Conlen

-----Original Message-----

From: owner-freebsd-performance@freebsd.org

[mailto:owner-freebsd-performance@freebsd.org]On Behalf Of D. J. Bernstein

Sent: Saturday, June 21, 2003 2:58 PM

To: freebsd-performance@freebsd.org

Subject: ten thousand small processes

FreeBSD 4.8. Test program: malloc(360); malloc(80); malloc(180); malloc(16); malloc(440); sleep(10); _exit(0). Compile statically.

The program ends up with 44KB RSS. Where is all that DRAM going? The program also ends up with 168KB VSZ. Where is all that VM going?

I don't care much about the 3-page text segment. But I do care about the 39 extra pages of VM, and the 8 extra pages of DRAM. There's no obstacle to having a small program fit into one page per process; two or three can be excused, but 39 is absurd. (Yes, I know that Solaris is worse.) At least 2 pages appear to be wasted by exit(), because it brings in a chunk of stdio, which uses 84 bytes of data and 316 bytes of bss. The libc implementors clearly don't care about 316 bytes of memory, so why don't they make those 316 bytes static? Why doesn't the compiler automatically merge some bss into data when that saves a page? Why can't I omit exit(), manually or automatically, when it's unreachable?

Furthermore, malloc() appears to chew up a whole new page of DRAM for each allocation, plus another page---is this counted in VSZ?---for an anonymous mmap. Would it really be that difficult to fit 1076 bytes of requested memory into the 3000-odd bytes available at the end of bss? I sure hope that there's some better explanation for the remaining 32 pages than ``Well, we decided to allocate 131072 bytes of memory for the stack,`` especially when I'm hard-limiting the stack to 4K before exec.

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

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"

freebsd-performance@freebsd.org mailing list

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

freebsd-performance: RE: ten thousand small processes

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