

Re: increased machine precision for FreeBSD apps?

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-07/0055.html>

From: David Schultz (*das_at_FreeBSD.ORG*)

Date: 07/15/04

Date: Thu, 15 Jul 2004 08:39:20 -0700

To: Jay Sern Liew <liew@jaysern.org>

On Sun, Jul 11, 2004, Jay Sern Liew wrote:

- > *This is probably more of an organizational issue than an architectural*
- > *one. In the event of running applications with heavy floating point*
- > *arithmetic (e.g. scientific computation apps for one, encryption*
- > *algorithms, compression, .. etc), would it be a good idea to decrease*
- > *rounding errors by increasing the data type size?*

Most programmers expect a float to be an IEEE 754 single-precision number, and they expect a double to be a double-precision number. On most modern FPUs, however, floats, doubles, and sometimes even long doubles take about the same number of clock cycles. The only good reason to use float is for things like genomics, where you're doing calculations on gigabytes of data, and the goal is to fit as much of it in memory as possible.

So here's how you handle the problem. If you want to use the largest type that's at least as big as a float, use C99's float_t instead. If you want the largest type that's at least as big as a double, use double_t. If you want the largest type possible, at the expense of having to emulate it on architectures such as PowerPC, use long double. If you want to run one of those scientific apps that phk says don't exist, use an arbitrary-precision arithmetic package such as Gnu MP, Paul Rouse's public domain clone thereof, QLIB, or Mathematica.

freebsd-arch@freebsd.org mailing list

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

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