Re: lines of code?

Source: http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2003−12/1564.html

---

From: goose (*ruse_at_webmail.co.za*)
Date: 12/12/03

Date: 12 Dec 2003 04:18:17 −0800

Chris Smith <cdsmith@twu.net> wrote in message
news:<MPG.1a42133afc6bc2b498980c@news.pop4.net>...
> Hi goose,
>
> Thanks for your reply. If you don't mind, I'm going to simplify your
> response by eliminating the part about embedded systems. That seems to
> be your area, and the same ideas apply in either case... but I'm
> guessing that for the majority of the readership, high−level libraries
> such as UNIX select() and other means of async I/O are going to be the
> way to go.

No problem.

>
> goose wrote:
> > Your "alternative" to multiple threads is to "jump back and forth
> > from one task to another [in] the same thread". I'd suggest that
> > if your /solution/ demands multiple threads and/or tasks, then go
> > ahead and use it, but dont try to implement a multiple−thread
> > solution in a single thread. that just wont work nicely.
>
> Just to clarify, we are talking about whether multithreading is ever
> justified on a uniprocessor, right? That's what I thought we're talking
> about. It sounds like you're saying that I should use multiple threads
> on a uniprocessor in some cases.

Nope. If your *solution* is *designed* as lots of seperate tasks,
then perhaps creating threads for each task is best. My contention
is that a lot of *designs* that calls for multiple threads
dont really get anything more than a single−threaded design would.

In other words, very many wizz−bang multi−threaded apps would be
no better in performance and ease of maintenance than single−threaded
apps (very often, there is a performance degradation for the multiple
threads).

> If you're saying that, then perhaps I
> just imagined the difference of opinion in the first place.

> *Nevertheless, you later said:*
>
> > *in a server environment, lets say a unix machine of some sort*
> > *(or even*