



Re: A chip too far? Where is your solution Mr Larkin?

Can't fault the idea of keeping the Ring0 protected kernel as small as possible, or giving users the least privileges needed to do their job.

But wasting performance for reliability though is never going to fly – not least because most times it would not have the desired effect.

Hardware enforced memory protection for threads on a timesliced CPU can be made every bit as reliable as giving each one a physical CPU just a SMOP.

3. Ultimately, a new multicore cpu architecture that exerts much more hardware control over the things that programmers tend to screw up.

You mean like enforcing bounds checking on array and pointer access?

Pointers have to be the first thing to go. And of course array bounds should be checked, but all that can do is crash a program, which was going to crash sooner or later anyhow.

Pointers are OK provided you can only trash your own memory space. Anything else and you should get squashed flat.

We are programming antique hardware in antique languages.

Speak for yourself. I program in the language that best fits the problem (or is dictated by the clients requirements). Sometimes both. The fastest way to develop really tricky algorithms is in a language where human errors and typos are mostly caught at compile time. The final version is then translated back into C (which is mostly what people seem to want). ISTR NAGLIB used this approach a very long time ago (when FORTRAN was the main language of choice for scientific computing).

I wish that a more robust language had won the language wars, but it is too late now to cry over spilt milk.

So, a hundred years from now, everybody will still be programming in C++? Writing billion-line programs with millions of bugs?

Something to look forward to.

Re: A chip too far? Where is your solution Mr Larkin?

Re: A chip too far? Where is your solution Mr Larkin?

John

.