

## Re: can anyone offer Lisp job?

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-08/msg02345.html>

---

- *From:* Kent M Pitman <[pitman@xxxxxxxxxxx](mailto:pitman@xxxxxxxxxxx)>
  - *Date:* Sun, 28 Aug 2005 18:51:26 GMT
- 

Ulrich Hobelmann <[u.hobelmann@xxxxxx](mailto:u.hobelmann@xxxxxx)> writes:

> Kent M Pitman wrote:  
>> I liked what US politician and wrestler Jess Ventura said about  
>> Government [no exact quote handy, so I'll paraphrase]: that government  
>> should only do for people what they cannot do for themselves. There's  
>> probably some fun we can have with that in the Common Lisp context,  
>> since I'm sure someone will say this argues for turning Common Lisp  
>> into scheme.  
>  
> Hehe :)  
>  
> Actually it's more about the kernel not providing anything that the  
> userland can do. I think the Lisp kernel isn't that much bigger than  
> the Scheme kernel. The libraries are just standardized, which is  
> definitely a good thing.  
>  
> The Scheme kernel lacks, for instance, a package system. It's hard to  
> do that in user-space, especially in a uniform way. But then you  
> might say that Scheme is an experimental language to try things like  
> hygienic macros (in different implementations), continuations,  
> different module systems, and different lighter-than-CLOS OO systems.

The hard thing for people to understand about CL is how much it was an exercise in budgets and resources, not in technical decisions.

I submitted a bunch of change requests you see in the X3J13 issue index in CLHS. A lot of mine got through. Periodically people would ask "why yours?" and I'd answer "because I wrote them up. if others did, we should look at those, too. if there are too many, we should prioritize them. but we shouldn't not do mine because of hypothesized other needs that no one has actually raised."

As I look back at code that I even thought was GOOD CODE in Maclisp, it looks TERRIBLE to me now. It's hard to impress on people how the notion of The Aesthetic evolves over time, but yet you're intimately familiar with it if you look to new industries like TV and Movies. Look at what used to pass for a good movie and what does now. Actually, the better way to say it was that there are some excellent

## Re: can anyone offer Lisp job?

old movies and some excellent old TV, but it happened more by accident or inspiration or art than by Science. The same is true of programming. There was cool work on programming. EVERYONE should read the original Algol 60 report, the original Scheme report, Steele's Lambda The Ultimate series, etc. There were brilliant things done back then. But it was less a matter of "regular practice" and more a matter of luck, that the right people were doing it. Nowadays, we've studied what we like and don't from that era and we have more refined notions that languages should have kernels, but that notion arises from the earlier ideas, like even Guy Steele's observation in CLTL1 that the compiler semantics and interpreter semantics of CL were the same. They weren't in Maclisp. In Maclisp, compiling code ran one processor over it and interpreting it another, and the whole issue of variable scope was handled VERY differently because there wasn't the notion of Language As King. The implementation was king, and the language definition followed from understanding the implementation.

(This, incidentally, is why I don't like the CLRFI model. Languages and specs should stand free of implementation. Having an implementation is an irrelevance to me when discussing languages. A language idea is either good or not in the abstract—someone's decision to implement it or not is one kind of proof it can be made, but it doesn't give credence to an idea. Good language ideas are, by nature, implementable. But there are many ways one can show something implementable short of creating an implementation. Anyway, I digress.)

But the notion that a kernel language was an important idea did not precede languages becoming large. No one envisioned languages GETTING large until it happened, and only then as an effect of bookkeeping did it seem sensible to approach that. Eulisp confronted this earlier, but even then they got side-tracked on the serious issue of whether the layering should be "implementation" or "spec". That is, the inner layer was often described to me as "scheme-like" and then I'd always ask "oh, so Eulisp is a superset of Scheme" and then I'd be told by some in the project "no, the kernel isn't a subset of the language, it's an implementation substrate for the language". But that seemed to vary depending on whether you talked to the guys implementing the kernel or the layered parts. Everyone felt their part was central, it seemed to me, and I suppose ultimately something came of all that, but it isn't a major player in the world lisp market today, and I suspect at least to some degree because it didn't have this detail down. Not that it wasn't a good idea, but the details matter.

The time when CL was first asked to be modular (different than small, there were people who wanted small much earlier, but they were making what appeared to be more drastic requests like "don't address that area" not "do it in modules") was late in the X3J13 standards process, around 1988 or later (1988 being the theoretical "feature freeze" for the 1994 spec, though we made "essential changes" up to about 1992). There was pressure to make some components optional, too, but that again was not presented

Re: can anyone offer Lisp job?

## Re: can anyone offer Lisp job?

as a coherent modular componentry—more of a "do this part if you want and leave out other parts if you don't want". That was still kind of whimsy from a theoretical design point of view, and didn't explain how things would plug-and-play if one implementation wanted CLOS and all its objects based on it, but others wanted no CLOS and didn't want anything to hint that CLOS was there. That isn't a kernel issue, it's a mess.

The first rational proposal for a layered Common Lisp arrived late in the public review process. It came from Paul Robertson (author of Robertson Common Lisp, which later became Symbolics CLOE for the i386). Paul did the hard work of separating all the parts of the language into different parts and suggested some theory of layering. But it was controversial in its details and it was too late in the process for him to convince everyone. There was an echo of the same fear that we'd heard from earlier teachers that if we made a "teaching subset" it would work for one teacher but not another. Like "prayer in school", it sounds good in principle but when you get to details, not everyone agrees on what's in and what's out.

The sense among the X3J13 committee, as I recall it (and I'm obliged to say that no one can speak officially for it on such matters—I'm just offering my personal opinion/recollection) was that layering could be retrofitted by implementations, and so the decision to leave out layering was not fatal to the ultimate success of CL.

Now, that doesn't mean implementations have or will. And you should be careful to understand that I'm not saying that the failure to DO layering won't ultimately be fatal. Maybe it will, but I think that needs to be shown. But my point is that the failure to do it is not just on the designers. Nothing in the spec says you can't subset things, you just have to identify subsets. Nothing says you can't do precisely as Paul Robertson suggested and make some tag for REQUIRE that if you use you get the whole language even though there were ways to get parts of it under other situations.

And the presence of those freeware things that I lament so much give me moral leverage to say back to each and every person who whines "but I'm not an implementor, I can't do that" that "you could be an implementor, just pick up a freeware CL and start hacking. build something that shows the kernel if that's what you think is economically worth doing". If you find you have a day job and mouths to feed and not enough time to do it because no one is paying for kernelness, then maybe you'll see the problem the vendors have now. Or if you find yourself beating the vendors in sales because now you have CL done right and it's all anyone ever wanted, you'll have done a wonderful thing for everyone.

Then again, you might try, and it might be cool, and still it might not affect things. Look at how cool CMU's python compiler is and how many people have rushed to prefer that as a result.

.

Re: can anyone offer Lisp job?

- **Follow-Ups:**
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Juho Snellman
  
- **References:**
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Robert Maas, see <http://tinyurl.com/uh3t>
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* jbo
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* alex . gman
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* David Steuber
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Kent M Pitman
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Bulent Murtezaoglu
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Kent M Pitman
  - ◆ **Re: can anyone offer Lisp job?**
    - ◇ *From:* Ulrich Hobelmann
  
- Prev by Date: **Re: modifying array access syntax**
- Next by Date: **Re: novice: mapcan use?**
- Previous by thread: **Re: can anyone offer Lisp job?**
- Next by thread: **Re: can anyone offer Lisp job?**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**