PANEL ON THE CONCEPT OF EXTENSIBILITY

Chaired by Peter Wegner, Cornell University.

This panel was composed of Cheatham, Christensen, McIlroy, and Nicholls. It came at the end of the day, and the discussion sometimes became rather specialized. Remarks of general interest follow.

M.C. Harrison: I want to make two comments and one short commercial. First of all, a trivial point about the word that we are all using. "Extensibility" smacks too much of "extensive" to me, I prefer extendibility.

The second point is that any programming language in which programs and data are essentially interchangable can be regarded as an extendible language. I think this can be seen very easily from the fact that Lisp has been used as an extendible language for years.

Now my commercial: we have a "son of Lisp" extendible language which we have been implementing at New York University over the past few months. This is based on an extendible Lisp system that has vectors and strings. You can do the same sort of thing with vectors as you can with lists, except that you use square brackets instead of round brackets and you can index the elements. We put a translator on the front of this to make it look like Algol. I guess it is in the direction of Lisp 2, but the translator takes about two thousand word on the 6600; it is a very simple translator. The programmer can add prefix or infix operators, he can specify left and right precedences of those operators and he can also define macros that are applied to the results of the precedence analysis. Altogether, we can get this Algol-like language with virtually nothing built into the translator except the operator precedences and the macros.

McIlroy: I think that not all the languages we saw today are "sons of Algol". In fact, I'd very boldly classify their Designers into two categories: the anarchists and the fascists.

We have two anarchists in Irons and Standish; anarchists in the sense that these languages, particularly Irons' language, seem to be devoid of data types, preconceptions about scope, and persistence of data. They have, in some sense, the same capabilities that a computer has naked, before it has been reined in to run as an Algol machine.

For extensions which are trivial, but perhaps more desirable to the user, the fascists (who say "anything you want to do, as long as it's my way, is all right") probably have the edge. For interesting experiments in language, I think I would turn more to the anarchists, who don't have so many preconceptions about how we compute. It is the fascists who are the sons of Algol.

J. A. Feldman: I have two interrelated comment/questions. The first one has to do with this extensibility game. I thought I understood it before I came in here, and I will first tell you what I think has happened here. It seems to have now become "all useful things to do in helping to write systems programs" and, therefore, no one could possibly be against it. That is, we have heard about editors, compiler-compilers, and all these things are part of the extensible language game. If that is true, then I think we ought to get a new word for what I used to think extensible languages were--which is that one defined additions to the language almost wholly in terms of the existing language without any direct reference to the compiler, that is in a macro-like fashion. Now I notice that Tim Standish and Ned Irons are now looking at things I would have called translator-writer systems techniques and saying "Yes, we will go back and see what happens in Phase 2."

This leads me to my second comment. I have been able until now to at least convince myself that what has been called extensible languages can be dismissed in about three sentences, which go like this: If you are making a non-trivial extension to the language you are changing the compiler in an important way. If you are doing that, why do you try to do it in an non-procedural, macro-like mystical fashion, why don't you write the compiler in a coherent manner, and then the person who is going to change the compiler goes and does that. If the compiler is written in the source language that is not unreasonable.