WIRTH83

Wirth, N.; The Programming Language Modula-2; Institut fur Informatik, Report Nr. 36 IfI ETH Zurich, 1980, pp. 1-25. Also appears as pp. 139-170 of Wirth, N.; Programming in Modula-2; New York: Springer-Verlag, 1983.

Modula-2 is a programming language developed between 1977 and 1979 by Niklaus Wirth as a successor to Modula and Pascal. Modula-2 includes most of the features (good and bad) of Pascal, with some additional ideas on modules most of which are based on the Mesa language developed at Xerox PARC where Wirth visited in 1976.

Modula-2 adds a number of features to Pascal. First, "A more systematic syntax" which means that every structure starting with a keyword ends with a keyword. The result is that a Modula-2 program has fewer BEGINs than Pascal, but many more ENDs. There are some other minor changes in the syntax. For example, each clause of a CASE statement ends with a "|" instead of a ";" and there may be an "ELSE" clause in a CASE; records can have more than one variant part; "POINTER TO Foo" is used instead of "^Foo"; "AND" and "OR" exit as soon as the final value is known (rather than always evaluating all clauses); constant expressions are supported as are array parameters with unspecified bounds in procedures; and there is a CARDINAL type which is for positive integers. Next, there are low level features that are supposed to make it easier to build system software. These include the ability to easily get around the rigid type rules, the ability to do pointer arithmetic, and a built-in function to get the address of a variable.

Procedure types, variables and parameters were also added to the language. Whereas this is a very useful feature, the syntax does not seem to fully support it. A function with no parameters can be invoked by "Foo" (rather than "Foo()" in Mesa) which seems to make it ambiguous for a function that returns a function whether "Z := Foo" means Z gets the function Foo or the function that Foo returns.

The most important addition to Modula-2 is the concept of a Module. Modules are scoping mechanisms that allow certain variables, types, constants and procedures to be explicitly exported while others are protected from outside use. Modules also have explicit imports so it is clear exactly what the module uses. To avoid name conflicts, identifiers may be qualified with the name of the module that they come from. Pointers can also be exported opaquely so that clients can pass them to routines but cannot dereference them. Modules can therefore be used to provide abstraction mechanisms and information hiding as required in good structured programming techniques. Associated with modules is the ability to have separate compilation. The export section of a module can be copied into a separate "definition" file which is imported by other modules using that module. The implementation can then be compiled separately.

Modula-2 has eliminated the powerful I/O primitives READLN and WRITELN from Pascal. With modules, the desired read and write procedures can be implemented separately and used in all application programs, but the ability to mix outputs or inputs of different types is lost. This will probably be fairly inconvenient for many programmers.

Finally, Modula-2 provides multiple processes. These are implemented as loosely coupled co-routines with explicit programmer-defined transfer back and forth. It appears to be a very low level mechanism since when creating a new process, the creator must know exactly how much memory to allocate for the new process's stack. Processes in Modula-2 should be easy to implement, however, and the more complex schemes such as monitors and signals can be written using it. Wirth also claims that the simple mechanism is often all that is needed, especially for device drivers for simple operating systems such as those for personal computers.

Although little in Modula-2 seems to be original, it seems to be a better and more powerful language than Pascal. Since many groups that use Pascal need to extend it in non-standard ways to add features like those in Modula-2, the wide acceptance of Modula-2 may allow for much better standardization of software industry-wide. Modula-2 does have some problems, but, in general, it seems to be powerful enough to be useful without extensions but simple enough to implement easily, so it is possible that it may become widely used on personal machines and elsewhere.