WG14/N367
X3J11/94-052

Future Directions For C

Francis Glassborow, ACCU Chair
64 Southfield Road
Oxford, OX4 1PA
UK
francis@robinton.demon.co.uk

In considering specifications for the work undertaken to develop the next version of ISO C we should take into account the 'Spirit of C'.

As I understand it, one primary intention shared by all those involved in development to date has been to provide effective mechanisms for programmers to express their solutions in terms that are as portable as possible across a wide range of machine architectures. At the same time the resulting code should be efficient while supporting the maxim that 'programmers know what they are doing.'

If the next version of C is no more than a number of relatively minor developments and additions based on the same 'ideal machine' that the current version is based on it will fail to support the 'Spirit of C' becuase it will not support the programmers of the early decades of the twenty-first century express their intentions to the machine architectures that will then be common.

By then, array processors and other forms of parallel processors (both SIMD and MIMD) will be generally available. In such an environment C should provide language elements to enable programmers to express their intentions and responsibilities for using multiple threads. Currently all C must be compiled to behave 'as if' it is running as a single thread on a single processor. Quite apart from this placing a substantial burden on the flow analysis capacity of a compiler compiling to use more extensive resources it also confines programmers to using single thread algorithms which may be poor choices when parallel processors are available.

C is sometimes described as a powerful, high level, portable assembler. There is more than a modicum of truth in this description which explains its power in a number of problem domains that it has made almost exclusively its own. If C is to continue to serve as a 'portable, high-level assembler' it will need to add support for the coming generations of machine architectures. Time spent on largely cosmetic improvements to the language will be time wasted because such a language will not meet the needs of programmers and so will justifiably die.