Document: WG14 N1509

Optimizing away infinite loops

Submitter: Douglas Walls (USA)
Submission Date: 2010-09-02
Related documents: N1349
Subject: Optimizing away infinite loops

Paper N1349 Parallel memory sequencing model proposal ... added a new paragraph 6.8.5p6 which is still present in the latest C1X draft N1494:

> 6 An iteration statement that performs no input/output operations, does not access volatile objects, and performs no synchronization or atomic operations in its body, controlling expression, or (in the case of a for statement) its expression-3, may be assumed by the implementation to terminate.[154)]
>
> 154) This is intended to allow compiler transformations such as removal of empty loops even when termination cannot be proven.

One consequence of the memory model is that you cannot advance modifications to an object across potentially nonterminating loops, since that may introduce data races. Stores that wouldn't really have happened are now performed. This seems to disable some optimizations, for example a compiler cannot automatically parallelize the outer loop of a loop nest without a guarantee that the inner one always terminates. So C++ attempted to clarify the situation by adding the equivalent of 6.8.5p6 which C1X draft adopted as part of the memory sequencing model.

It has been argued that this change was really an attempt to preserve the status quo. That the rules of C99 subclause 5.1.2.3p5 do not make

it clear that an compiler that translates `int main() { for(;;); return 0;}` into `int main() { return 0; }` is not conforming.

I content that a great many C90/C99 programs rely on entrance to an unending loop to proceed no further (within that thread). Those programs are relying on the implementation to implement the semantics of C99 6.8.5p4: "An iteration statement causes a statement called the loop body to be executed repeatedly until the controlling expression compares equal to 0." I don't believe there are any programs that rely on a compiler to translate `int main() { for(;;); return 0;}` into `int main() { return 0; }`. So I think it is pretty clear that this change affects the (observable and desired) behavior of strictly conforming programs. Thus introducing a serious incompatibility between C99 and C1X.

Though I am sympathetic to the intent, I do not have any alternatives to suggest to address that intent. So, I'm proposing with this paper that we remove paragraph 6 of subclause 6.8.5 and it's accompanying footnote.