GOMP — The GNU OpenMP Project

Requirements

written by **Scott Robert Ladd** <coyote@coyotegulch.com>

Work on this document was sponsored by **CodeSourcery, LLC**.
(http://www.codesourcery.com)

The GOMP project is developing an implementation of OpenMP for the C, C++, and Fortran 95 compilers in the GNU Compiler Collection. As part of the GNU Project, GOMP will simplify parallel programming for all GNU system variants. This effort operates in an open environment to attract developers and ensure applicability across multiple architectures and applications.

This document outlines the requirements for implementing the OpenMP Standard in GCC.

Justification

Traditionally, programmers have used architecture-specific methods to effectively program tightly-parallelized computers — high band-width clusters, SMP machines, or multi-core processors. Parallel programming has thus been a time-consuming and arcane task.

OpenMP offers a simple way of exploiting parallelism without interfering with algorithm design; an OpenMP program compiles and operates correctly in both parallel and serial execution environments. Using OpenMP's directive-based parallelism also simplifies the act of converting existing serial code to efficient parallel code.

To remain relevant, free software development tools must support emerging technologies. By implementing OpenMP, GOMP will provide a simplified syntax tools for creating software targeted at parallel architectures. OpenMP's platform-neutral syntax meshes well with the portability goals of GCC and other GNU projects.

Standards

These are the documents, standards, and philosophies that will guide GOMP development.

- GOMP will implement version 2.0 of the OpenMP standards for Fortran C and C++. The documents for these standards can be obtained from http://www.openmp.org.

- GOMP will build OpenMP on the GCC 4.x code base; it will not be backported to earlier versions of GCC. Development will take place in a separate branch, gomp-branch, to avoid complicating mainline development.

- All GOMP code will adhere to GNU coding conventions and standards. Work will be undertaken in the open community spirit of free software.

Effects

While this is not a design document, it is wise to understand, in the beginning, how OpenMP will affect the compiler. Thus the discussion here is general; a more specific plan will be outlined in a separate design document.

- Implementing OpenMP will require changes to the C, C++, and Fortran front ends. These changes will allow those front ends to semantically parse OpenMP directives — pragmas in the cases of C and C++, special comments in Fortran 95. All effort will be made to unify these modification to limit duplication of effort. Work on C and C++ will account for proposals for a unified parser for those two languages.

- Change will be required to the middle end of the compiler. OpenMP directives cause the compiler to analyze program code (e.g., determining local and shared variables) and insert new code and or rearrange existing code accordingly; in some cases, how code is reorganized is dependent on the threading model in use.

- The GOMP release will include a support library, libgomp, that implements both OpenMP-defined and GCC-internal functions.

Goals

GOMP has several primary goals that define the minimum requirements for including OpenMP in mainline development.

- The foremost goal is correctness; programs compiled with OpenMP must operate as expected according to programming language and OpenMP standards.

- GOMP must not cause any regressions in the compilers or the test suites. The current test suite must produce the same results with and without OpenMP enabled.

- An additional test suite is necessary to verify both the completeness and correctness of GOMP's OpenMP. Additional performance tests may be in order, although this is a secondary goal.

- GOMP will be well documented, such the other developers will understand the how OpenMP affects GCC, and to ensure future maintainability.

- GOMP will not be released without support for x86_64 and i686 SMP and NUMA architectures running Linux and POSIX threads. This is the minimum level of platform support required for GOMP to be merged into mainline GCC.

Secondary Goals

The following are important goals that do not necessarily affect whether or not GOMP is accepted into mainline GCC development. These can also be considered ongoing targets for future development.

- An effort should be made to ensure the OpenMP programs run quickly and without using excess system resources. Future versions of GOMP will work to improve performance.

- In keeping with GCC's cross-platform spirit, GOMP will implement OpenMP for as many architectures as is practical, given the expertise of developers and the availability of hardware.

Operation

- OpenMP directives will act as comments (C95) or unimplemented pragmas (C/C++) unless the **-fopenmp** option is specified on the compiler command line. When **-fopenmp** is used, the compiler will generate parallel code based on the OpenMP directives encountered.

- By default, the compiler will define the option **-fno-openmp**.

- If **-fopenmp** is not specified, the compiler will generate a serial program that operates as if the OpenMP directives were non-existent.

- Linking to the OpenMP support library will be automatic if **-fopenmp** is specified for linking.

- GOMP will use the threading model specified by the **–enable-thread=xxx** configuration option.

end # #