```
int MPI::Comm::Get_size() const
{
  return pmpi_comm.Get_size();
}
```

Example 16.10 `mpi_profile.cc`, to be compiled into `libpmpi.a`.

```
int MPI::Comm::Get_size() const
{
  // Do profiling stuff
  int ret = pmpi_comm.Get_size();
  // More profiling stuff
  return ret;
}
```

(*End of advice to implementors.*)

16.2 Fortran Support

16.2.1 Overview

ticket103.

[Fortran 90 is the current international Fortran standard. MPI-2 Fortran bindings are Fortran 90 bindings that in most cases are "Fortran 77 friendly." That is, with few exceptions (e.g., `KIND`-parameterized types, and the `mpi` module, both of which can be avoided) Fortran 77 compilers should be able to compile MPI programs.

Rationale. Fortran 90 contains numerous features designed to make it a more "modern" language than Fortran 77. It seems natural that MPI should be able to take advantage of these new features with a set of bindings tailored to Fortran 90. MPI does not (yet) use many of these features because of a number of technical difficulties. (*End of rationale.*)

MPI defines two levels of Fortran support, described in Sections 16.2.3 and 16.2.4. A third level of Fortran support is envisioned, but is deferred to future standardization efforts. In the rest of this section, "Fortran" shall refer to Fortran 90 (or its successor) unless qualified.]

The Fortran MPI-2 language bindings have been designed to be compatible with the Fortran 90 standard (and later). These bindings are in most cases compatible with Fortran 77, implicit-style interfaces.

Rationale. Fortran 90 contains numerous features designed to make it a more "modern" language than Fortran 77. It seems natural that MPI should be able to take advantage of these new features with a set of bindings tailored to Fortran 90. MPI does not (yet) use many of these features because of a number of technical difficulties. (*End of rationale.*)

MPI defines two levels of Fortran support, described in Sections 16.2.3 and 16.2.4. In the rest of this section, "Fortran" and "Fortran 90" shall refer to "Fortran 90" and its successors, unless qualified.

the & operator and later referencing the objects by way of the pointer is an integral part of the language. A C compiler understands the implications, so that the problem should not occur, in general. However, some compilers do offer optional aggressive optimization levels which may not be safe.

16.2.3 Basic Fortran Support

Because Fortran 90 is (for all practical purposes) a superset of Fortran 77, Fortran 90 (and future) programs can use the original Fortran interface. The following additional requirements are added:

1. Implementations are required to provide the file `mpif.h`, as described in the original MPI-1 specification.
2. `mpif.h` must be valid and equivalent for both fixed- and free- source form.

   Advice to implementors. To make `mpif.h` compatible with both fixed- and free-source forms, to allow automatic inclusion by preprocessors, and to allow extended fixed-form line length, it is recommended that requirement two be met by constructing `mpif.h` without any continuation lines. This should be possible because `mpif.h` contains only declarations, and because common block declarations can be split among several lines. To support Fortran 77 as well as Fortran 90, it may be necessary to eliminate all comments from `mpif.h`. (*End of advice to implementors.*)

16.2.4 Extended Fortran Support

Implementations with Extended Fortran support must provide:

1. An `mpi` module
2. A new set of functions to provide additional support for Fortran intrinsic numeric types, including parameterized types: MPI_SIZEOF, MPI_TYPE_MATCH_SIZE, MPI_TYPE_CREATE_F90_INTEGER, MPI_TYPE_CREATE_F90_REAL and MPI_TYPE_CREATE_F90_COMPLEX. Parameterized types are Fortran intrinsic types which are specified using `KIND` type parameters. These routines are described in detail in Section 16.2.5.

Additionally, high-quality implementations should provide a mechanism to prevent fatal type mismatch errors for MPI routines with choice arguments.

The `mpi` Module

ticket103. An MPI implementation must provide a module named `mpi` that can be [`USE`]`use`d in a Fortran 90 program. This module must:

- Define all named MPI constants
- Declare MPI functions that return a value.

An MPI implementation may provide in the `mpi` module other features that enhance the usability of MPI while maintaining adherence to the standard. For example, it may: