

**MPI\_PCONTROL(level, ...)**

IN            level                            Profiling level

int MPI\_Pcontrol(const int level, ...)

MPI\_PCONTROL(LEVEL)

INTEGER LEVEL

void MPI::Pcontrol(const int level, ...)

MPI libraries themselves make no use of this routine, and simply return immediately to the user code. However the presence of calls to this routine allows a profiling package to be explicitly called by the user.

Since MPI has no control of the implementation of the profiling code, we are unable to specify precisely the semantics that will be provided by calls to **MPI\_PCONTROL**. This vagueness extends to the number of arguments to the function, and their datatypes.

However to provide some level of portability of user codes to different profiling libraries, we request the following meanings for certain values of level.

- **level==0** Profiling is disabled.
- **level==1** Profiling is enabled at a normal default level of detail.
- **level==2** Profile buffers are flushed. (This may be a no-op in some profilers).
- All other values of **level** have profile library defined effects and additional arguments.

We also request that the default state after **MPI\_INIT** has been called is for profiling to be enabled at the normal default level. (i.e. as if **MPI\_PCONTROL** had just been called with the argument 1). This allows users to link with a profiling library and obtain profile output without having to modify their source code at all.

The provision of **MPI\_PCONTROL** as a no-op in the standard MPI library allows them to modify their source code to obtain more detailed profiling information, but still be able to link exactly the same code against the standard MPI library.

## 14.4 Examples

### 14.4.1 Profiler Implementation

Suppose that the profiler wishes to accumulate the total amount of data sent by the **MPI\_SEND** function, along with the total elapsed time spent in the function. This could trivially be achieved thus

```
static int totalBytes = 0;
static double totalTime = 0.0;

int MPI_Send(void* buffer, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_comm comm)
{
    double tstart = MPI_Wtime();      /* Pass on all the arguments */
    int extent;
```