NAME

MPIL_Trace_off − LAM/MPI-specific function to disable run-time tracing

SYNOPSIS

#include <mpi.h>
int MPIL_Trace_off(void)

NOTES

These functions give the application the flexibility to generate traces only during certain interesting phases of the application's execution. This technique can considerably reduce the size of the trace files and burden of displaying them.

Both functions are collective over the *MPI_COMM_WORLD* communicator. In typical usage, the *-toff* option of *mpirun* (1) would be used to enable tracing, but start with the runtime switch in the off position. At the beginning of an interesting phase of program execution, *MPIL_Trace_on* would be called. *MPIL_Trace_off* would be called after the interesting phase. Tracing can be turned on and off many times. Each period of tracing eventually forms a trace segment in the trace file extracted by *lamtrace* (1). If the on/off functions are never used and tracing is enabled with the *-ton* option of *mpirun* (1), a single trace segment is produced.

The on/off functions have no effect if tracing is not enabled by *mpirun* (1) with either the *-ton* or *-toff* switches. Thus, an application can be littered with these functions but run without trace collection and very little additional overhead due to the no-operation function calls.

This is a LAM/MPI-specific function and is intended mainly for debugging. If this function is used, it should be used in conjunction with the *LAM_MPI* C preprocessor macro

```
#if LAM_MPI
MPIL_Trace_off();
#endif
```

LIMITATIONS

After the volume of generated traces exceeds a preset limit, the oldest traces are discarded in favour of new traces. Avoiding discarded traces is further incentive to use *MPIL_Trace_on* and *MPIL_Trace_off* .

NOTES FOR FORTRAN

All MPI routines in Fortran (except for *MPI_WTIME* and *MPI_WTICK*) have an additional argument *ierr* at the end of the argument list. *ierr* is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the *call* statement.

All MPI objects (e.g., *MPI_Datatype* , *MPI_Comm*) are of type *INTEGER* in Fortran.

ERRORS

If an error occurs in an MPI function, the current MPI error handler is called to handle it. By default, this error handler aborts the MPI job. The error handler may be changed with *MPI_Errhandler_set* ; the predefined error handler *MPI_ERRORS_RETURN* may be used to cause error values to be returned (in C and Fortran; this error handler is less useful in with the C++ MPI bindings. The predefined error handler *MPI::ERRORS_THROW_EXCEPTIONS* should be used in C++ if the error value needs to be recovered). Note that MPI does *not* guarantee that an MPI program can continue past an error.

All MPI routines (except *MPI_Wtime* and *MPI_Wtick*) return an error value; C routines as the value of the function and Fortran routines in the last argument. The C++ bindings for MPI do not return error values; instead, error values are communicated by throwing exceptions of type *MPI::Exception* (but not by default). Exceptions are only thrown if the error value is not *MPI::SUCCESS* .

Note that if the *MPI::ERRORS_RETURN* handler is set in C++, while MPI functions will return upon an error, there will be no way to recover what the actual error value was.

MPI_SUCCESS
- No error; MPI routine completed successfully.

MPI_ERR_OTHER
- Other error; use *MPI_Error_string* to get more information about this error code.

SEE ALSO

MPI_Trace_on, mpirun

LOCATION

mpil_trace.c