Error Handling for Libraries

William Gropp
Goals

- In general, error messages should
  - detailed
  - specific to the instance that caused the error
  - uniform in language
- In software libraries
  - Errors should be indicated by an error return or exception
  - The error return value should not require special treatment
    - for example, the user should be able to ignore it and not have to free it
- Messages and Codes
  - Documentation can provide far more detailed information about user and system errors than is appropriate for an error message
- Don’t forget Internationalization
  - Unicode
- Most of all, it must encourage use
  - The perfect system is useless if programmers won’t use it
Existing Tools for Error Messages

• National Language Support (NLS) tools
  – Good: Works with a message catalog (helps unify messages)
  – Widely available (part of Xopen/Unix)
  – Bad: Messages are all generic (“syntax error” not “variable foo is undeclared”)
  – Messages identified by a number in a routine call

• Message Format (msgfmt) tools
  – Good: message ids are text, not integers (see man -3i gettext on Solaris)
  – Bad: Messages are all generic
  – Bad: Ids not usable in libraries

• In both cases, message can contain formatting instructions, but then can’t be used directly with an id (where is the data saved?)
Error Handling in MPI

- Errors in MPI cause the MPI implementation to return an error code which is an integer.
- Each error code belongs to an error class. Example classes are MPI_ERR_ARG and MPI_ERR_RANK.
- An error code may contain additional information beyond the class.
- The routine
  \[
  \text{MPI\_ERROR\_STRING}( \text{int \ code}, \text{char *message}, \text{int *msglen} )
  \]
  converts a code into a character message. The routine
  \[
  \text{MPI\_ERROR\_CLASS}( \text{int \ code}, \text{int *class} )
  \]
  converts a code to a class.
- Errors invoke an error handler
  - MPI\_ERRORS\_ARE\_FATAL
  - MPI\_ERRORS\_RETURN
Error Handling in MPICH

- An error code in MPICH has the following parts:

<table>
<thead>
<tr>
<th>Sign</th>
<th>User</th>
<th>Ring_Index</th>
<th>Kind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>18</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

- Kind specifies a more specific, generic message within a class
- Ring_Index identifies a particular buffer containing a message that is specific to the instance that caused the error
- User is used to indicate a user-defined error class or code (part of MPI-2)
- Sign is zero to keep error codes non-negative
The Error Message Ring

- Instance specific messages are placed in a ring

<table>
<thead>
<tr>
<th>Ring Index</th>
<th>Specific Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Ring_Index is larger than the actual ring
  - Just use low bits

- Check full Ring_Index when accessing message
  - if does not match value in error code, message has been lost
    - Use generic message in that case
Placing Messages in the Source

- **MPIR_Err_setmsg** (class, code, routine_name, generic_string, instance_string, arguments, …)

- For example
  - `mpi_errno = MPIR_Err_setmsg( MPI_ERR_TOPOLOGY, MPIR_ERR_TOPO_TOO_LARGE, myname, "Topology size is larger than size of communicator", "Topology size %d is greater than communicator size %d", num_ranks, size );`

- Activating the appropriate error handler
  - `return MPIR_ERROR( comm_ptr, mpi_errno, myname );`
Creating the Message Catalogs

- **GetMsgCat**
  - Perl Script matches MPIR_Err_setmsg
    - Handles multiline uses
    - Generates
      - NLS style message catalog (for gencat)
      - mpierrstrings.h file containing all message strings
        » ensures that error messages can be generated even if there is a problem with the NLS system
        » No “NLS catalog not found” error messages 😊
Common Messages

• Some messages used in many routines, e.g.,
  – rank -3 is not a valid rank
  – variable buf is null

• These should have the exact same text everywhere

• Use
  – same text in all uses of MPIR_Err_setmsg
  – use (char *)0 to indicate default
    • default version of messages can be placed in a common file
Finding Mistakes

• Look for printf, puts and generate list
• Allow PRINTF, PUTS, etc. for non-user messages
  – Internal debugging messages
  – Language-independent output
• Definition of PRINTF, PUTS can be different from just printf, puts
  – Programs with no stdout/stderr (Windows)
  – Output in parallel applications
Further Improvements

- Make it easier to insert a new error message
  - Automatically detect and insert new error kinds
- Improve “Messages and Codes”
  - Automatically generate HTML etc.
  - Update more detailed descriptions