Projects and other things left to do

Coming Attractions

Some things that will probably be available in libg++ in the near future:

- Revamped C-compatibility header files that will be compatible with the forthcoming (ANSI-based) GNU libc.a
- A revision of the File-based classes that will use the GNU stdio library, and also be 100% compatible (even at the streambuf level) with the AT&T 2.0 stream classes.
- Additional container class prototypes.
- Generic Matrix class prototypes.
- A task package probably based on Dirk Grunwald's threads package.

Wish List

Some things that people have mentioned that they would like to see in libg++, but for which there have not been any offers:

- Class-based interfaces to Sun RPC using g++ wrappers.
- A method to automatically convert or incorporate libg++ classes so they can be used directly in Gorlen's OOPS environment.
- A class browser.

- A better general exception-handling strategy.
- Better documentation.

How to contribute

Programmers who have written C++ classes that they believe to be of general interest are encourage to write to dl at rocky.oswego.edu. Contributing code is not difficult. Here are some general guidelines:

- FSF must maintain the right to accept or reject potential contributions. Generally, the only reasons for rejecting contributions are cases where they duplicate existing or nearly-released code, contain unremovable specific machine dependencies, or are somehow incompatible with the rest of the library.
- Acceptance of contributions means that the code is accepted for adaptation into libg++. FSF must reserve the right to make various editorial changes in code. Very often, this merely entails formatting, maintenance of various conventions, etc. Contributors are always given authorship credit and shown the final version for approval.
- Contributors must assign their copyright to FSF via a form sent out upon acceptance. Assigning copyright to FSF ensures that the code may be freely distributed.
- Assistance in providing documentation, test files, and debugging support is strongly encouraged.

Extensions, comments, and suggested modifications of existing libg++ features are also very welcome.