Proposed Changes to Standard Exception Class in <stdexcept>

02/23/95

Owner: Albert A Slane

Contact: Albert A Slane
slane@rchland.vnet.ibm.com
IBM Rochester , MN

Problem with Current Class Exception

The following is the current exception base class from the Sept 20, 1994 Draft:

```
class exception {
public:
    exception(const string& what_arg);
    virtual ~exception();
    virtual string& what() const;
protected:
    exception();
private:
//  const string* desc;   exposition only
//  bool alloced;         exposition only
};
```

The current base class has two potential problems, both arising because potential memory allocation is done in the exception base class.

First, the ctor for class exception creates a new string from the passed argument what_arg. In general this is ok, but memory allocation can be expensive. In a real-time or embedded system this extra over head should be avoided if possible. Also, a memory allocation exception should be avoided while creating an exception if possible.

Secondly, the what() member function returns a reference to a new copy of the stored string. Again this is ok, but a copy is not necessarily what the client wants. For example, if the client just wants to print the string, a copy is not needed and the burden of deallocating the copy is placed on the client. This complicates client code and is inefficient.

Proposed Solution

Two changes can be made to the base class exception to fix the problems explained above and enhance the flexibility and efficiency of the exception hierarchy.

The following is the proposed version of class exception (changes marked with revision bar):

```
class exception {
public:
    exception(string& what_arg);
    exception(const string& what_arg);
    virtual ~exception();
    virtual const string& what() const;
protected:
    exception();
private:
//  const string* desc;  exposition only
//  bool alloced;        exposition only
};
```

First, a new ctor was added that takes a const_reference to a string. This ctor initializes desc to &what_arg and sets alloced to zero. This removes the memory allocation from the base class if this ctor is used, and allows the use of static const strings in derived classes (i.e. bad_alloc can use a static const string and avoid memory allocation). Note that the original ctor is also provided to allow derived classes the flexibility to use non-const strings.

Secondly, member function what() was changed to return a const_reference. This removes the memory allocation from the what() member function, and allows the client code to make a copy if one is needed. Also, the burden of deallocating the copy of the string is removed from the client unless the client specifically allocates a new copy.