GNU autosprintf, version 1.0

Formatted Output to Strings in C++

Bruno Haible

1 Introduction

This package makes the C formatted output routines (`fprintf` et al.) usable in C++ programs, for use with the `<string>` strings and the `<iostream>` streams.

It allows to write code like

```
cerr << autosprintf ("syntax error in %s:%d: %s", filename, line, errstring);
```

instead of

```
cerr << "syntax error in " << filename << ":" << line << ": " << errstring;
```

The benefits of the autosprintf syntax are:

- It reuses the standard POSIX printf facility. Easy migration from C to C++.
- English sentences are kept together.
- It makes internationalization possible. Internationalization requires format strings, because in some cases the translator needs to change the order of a sentence, and more generally it is easier for the translator to work with a single string for a sentence than with multiple string pieces.
- It reduces the risk of programming errors due to forgotten state in the output stream (e.g. `cout << hex;` not followed by `cout << dec;`).

2 The autosprintf class

An instance of class `autosprintf` just contains a string with the formatted output result. Such an instance is usually allocated as an automatic storage variable, i.e. on the stack, not with `new` on the heap.

The constructor `autosprintf (const char *format, ...)` takes a format string and additional arguments, like the C function `printf`.

Conversions to `char *` and `std::string` are defined that return the encapsulated string. The conversion to `char *` returns a freshly allocated copy of the encapsulated string; it needs to be freed using `delete[]`. The conversion to `std::string` returns a copy of the encapsulated string, with automatic memory management.

The destructor `~autosprintf ()` destroys the encapsulated string.

An `operator <<` is provided that outputs the encapsulated string to the given `ostream`.

3 Using autosprintf in own programs

To use the autosprintf class in your programs, you need to add

```
#include "autosprintf.h"
using gnu::autosprintf;
```

to your source code. The include file defines the class autosprintf, in a namespace called gnu. The 'using' statement makes it possible to use the class without the (otherwise natural) gnu:: prefix.

When linking your program, you need to link with libasprintf, because that's where the class is defined. In projects using GNU autoconf, this means adding 'AC_LIB_LINKFLAGS([asprintf])' to configure.in or configure.ac, and using the @LIBASPRINTF@ Makefile variable that it provides.