Defect Macros: Migrating to Standard C++ from Multiple non-Standard Dialects

29-Oct-1997

Rob Kennedy

A simple approach to building source code that can run with several different C++ dialects is described which uses preprocessor macros to isolate work-arounds for non-compliant compilers and uses centralized WWW documentation. The goal is to simplify use by many novice C++ programmers, and eliminate need for "re-discovery".

The Problem

- We are moving to Standard C++ (Kai C++ v3.2) from Gnu C++ v2.7, on SGI IRIX 6, Linux/Intel 2.0, and IBM AIX. We are maintaining support for SGI MipsPro 7.1nx on IRIX 6. We must do this carefully, maintaining support for all the above compilers for an interim period, with a few fundamental packages adapted first.
- Users are coding with a companion book describing Standard C++ .OR. they are coding using already written code in Gnu C++. Few know both dialects.
- Some differences between compilers are compile-time issues (usually easy to find if you use each compiler on each system, sometimes very difficult to understand), and some run-time issues (hard to find, no matter what).
- We lack central documentation of known defects and differences in the compilers.

Examples of Known Defects

Intro: http://www-cdf.fnal.gov/offline/zoom/basic_containers/kai_gcc_support.txt
Also: Follow-up slide on STL variations begin found by hand inspection

- "Defect" == Any non-compliance with the Dec. 1996 C++ Draft Standard.
- Stdlib header filenames are different.
- Stdlib classes are not in namespaces.
- Stdlib classes missing, substitute exists.
- Stdlib class methods are broken, missing, or just have different signatures.
- Some of these are examples are easy to work-around using the preprocessor, but we need to make it easy for ALL programmers to know this.
- Some "defects" require a style of coding to avoid, such as the lack of full support for namespaces.
- Some "defects" have no portable solution: RTTI.

Elements of Proposed Solution

- A common defect-oriented (instead of "__GNUG__ &&
  __GNUG_MINOR__ <= 7) nomenclature for macros, with Standard C++ always assumed to be the default.
- SoftRelTools defines the macros per system/compiler.
- A WWW document of the known defects and the recommended work-around macro blocks. Also document defects requiring a coding style to avoid or lacking a simple solution altogether.
- A Defects product which includes source code to test each work-around. This insures that the defect work-arounds are being tested for compilation (nightly rebuild), and users have access to this source.
- *Kai C++ v3.2d has been installed and tested on cdfsga.*
- SoftRelTools, *Trybos and Banks (done),* AC++, and Examples will be the first adapted... goal = this Friday.

Source Code Example 1

```
// File: Trybos/TRY_Bank_Type.hh
// Treat the unusual filename used by SGI CC for string
// Treat the lack of namespace support in Stdlib

#if defined(DEFECT_SGICC_NO_STDLIB_STRING)
#include <mstring.h>
#else // Standard C++
#include <string>
#endif

#if defined(DEFECT_NO_STDLIB_NAMESPACES)
// Do nothing
#else // Standard C++
using std::string ;
#endif
```

Source Code Example 2

```
// File: Trybos/TRY_Bank_Type.hh
// Treat old, deprecated stringstream class
// Treat lack of the Iostream forward declaration header

#if defined(DEFECT_OLD_STDLIB_STRINGSTREAM)
#define ostringstream ostrstream
class ostringstream ;
#else // Standard C++
#include  <iosfwd>
#endif

#if defined(DEFECT_NO_STDLIB_IOSFWD)
class ostream ;
#else // Standard C++
#include  <iosfwd>
#endif

// OMITTED #if defined(DEFECT_NO_STDLIB_NAMESPACES)
// FOR LACK OF SPACE ON SLIDE
```

Source Code Example 3

```
// File: Trybos/TRY_Bank_Type_vec2str.cc
// Treat the defect in Gnu C++ 2.7 stringstream::str()
// method in which it fails to return a proper string.

ostringstream type_string ; \\ ==> memory leak in G++ 2.7
string result ;

#if defined(DEFECT_GCC27_STRINGSTREAM_STR)
  type_stream << '\0' ;
  result = string(type_stream.str()) ;
#else \\ Standard C++
  result = type_stream.str() ;
#endif
```

Defects Identified So Far...

```
DEFECT_SGICC_NO_STDLIB_STRING -use <mstring.h> instead
DEFECT_GCC27_STRINGSTREAM_STR -use work-around in Ex.3

DEFECT_NO_STDLIB_IOSFWD - use bare forward class declare.
DEFECT_NO_STDLIB_NAMESPACES   -no hoisting from std::
DEFECT_OLD_STDLIB_IOSTREAM    -different set of filenames
DEFECT_OLD_STDLIB_STRINGSTREAM-different class used
DEFECT_OLD_STDLIB_STDC_HEADERS-no extern "C", filenames
DEFECT_OLD_STDLIB_STL_HEADERS -different set of filenames
DEFECT_OLD_STL_COUNT          -different signature

// DEFECT macros for vector::resize(), at() missing?
// DEFECT macro for ostream operator<<(pointer_type)?
// Even so, we cannot avoid adapting the compiler headers
// where possible to be closer to standard. For example,
// G++ string class on Linux has instantiation glitch,
// solution = update a file on Linux from cdfsga G++.
```