G++ 2.91.57，cygnus\cygwin-b20\include\g++**std\straits.h** 完整列表

```
// Character traits template for the -*- C++ -*- string classes.
// Copyright (C) 1994 Free Software Foundation

// This file is part of the GNU ANSI C++ Library.  This library is free
// software; you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the
// Free Software Foundation; either version 2, or (at your option)
// any later version.

// This library is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this library; see the file COPYING.  If not, write to the Free
// Software Foundation, 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.

// As a special exception, if you link this library with files
// compiled with a GNU compiler to produce an executable, this does not cause
// the resulting executable to be covered by the GNU General Public License.
// This exception does not however invalidate any other reasons why
// the executable file might be covered by the GNU General Public License.

// Written by Jason Merrill based upon the specification by Takanori Adachi
// in ANSI X3J16/94-0013R2.

#ifndef __STRING_CHAR_TRAITS__
#define __STRING_CHAR_TRAITS__

#ifdef __GNUG__
// For string_char_traits <char>
#pragma interface "std/straits.h"
#endif

#include <cstddef>

extern "C++" {
template <class charT>
struct string_char_traits {
  typedef charT char_type; // for users to acquire the basic character type

  // constraints

  static void assign (char_type& c1, const char_type& c2)
    { c1 = c2; }
  static bool eq (const char_type& c1, const char_type& c2)
    { return (c1 == c2); }
```

```
static bool ne (const char_type& c1, const char_type& c2)
  { return !(c1 == c2); }
static bool lt (const char_type& c1, const char_type& c2)
  { return (c1 < c2); }
static char_type eos () { return char_type(); } // the null character
static bool is_del(char_type a) { return 0; }
// characteristic function for delimiters of charT

// speed-up functions

static int compare (const char_type* s1, const char_type* s2, size_t n)
  {
    size_t i;
    for (i = 0; i < n; ++i)
       if (ne (s1[i], s2[i]))
         return lt (s1[i], s2[i]) ? -1 : 1;

    return 0;
  }

static size_t length (const char_type* s)
  {
    size_t l = 0;
    while (ne (*s++, eos ()))
       ++l;
    return l;
  }

static char_type* copy (char_type* s1, const char_type* s2, size_t n)
  {
    for (; n--; )
       assign (s1[n], s2[n]);
    return s1;
  }

static char_type* move (char_type* s1, const char_type* s2, size_t n)
  {
    char_type a[n];
    size_t i;
    for (i = 0; i < n; ++i)
       assign (a[i], s2[i]);
    for (i = 0; i < n; ++i)
       assign (s1[i], a[i]);
    return s1;
  }

static char_type* set (char_type* s1, const char_type& c, size_t n)
  {
    for (; n--; )
```

```
        assign (s1[n], c);
      return s1;
    }
};

class istream;
class ostream;
#include <cctype>
#include <cstring>

struct string_char_traits <char> {
  typedef char char_type;

  static void assign (char_type& c1, const char_type& c2)
    { c1 = c2; }
  static bool eq (const char_type & c1, const char_type& c2)
    { return (c1 == c2); }
  static bool ne (const char_type& c1, const char_type& c2)
    { return (c1 != c2); }
  static bool lt (const char_type& c1, const char_type& c2)
    { return (c1 < c2); }
  static char_type eos () { return 0; }
  static bool is_del(char_type a) { return isspace(a); }

  static int compare (const char_type* s1, const char_type* s2, size_t n)
    { return memcmp (s1, s2, n); }
  static size_t length (const char_type* s)
    { return strlen (s); }
  static char_type* copy (char_type* s1, const char_type* s2, size_t n)
    { return (char_type*) memcpy (s1, s2, n); }
  static char_type* move (char_type* s1, const char_type* s2, size_t n)
    { return (char_type*) memmove (s1, s2, n); }
  static char_type* set (char_type* s1, const char_type& c, size_t n)
    { return (char_type*) memset (s1, c, n); }
};

#if 0
#include <cwctype>
struct string_char_traits <wchar_t> {
  typedef wchar_t char_type;

  static void assign (char_type& c1, const char_type& c2)
    { c1 = c2; }
  static bool eq (const char_type & c1, const char_type& c2)
    { return (c1 == c2); }
  static bool ne (const char_type& c1, const char_type& c2)
    { return (c1 != c2); }
  static bool lt (const char_type& c1, const char_type& c2)
    { return (c1 < c2); }
```

```
  static char_type eos () { return 0; }
  static bool is_del(char_type a) { return iswspace(a); }

  static int compare (const char_type* s1, const char_type* s2, size_t n)
    { return wmemcmp (s1, s2, n); }
  static size_t length (const char_type* s)
    { return wcslen (s); }
  static char_type* copy (char_type* s1, const char_type* s2, size_t n)
    { return wmemcpy (s1, s2, n); }
  static char_type* set (char_type* s1, const char_type& c, size_t n)
    { return wmemset (s1, c, n); }
};
#endif
} // extern "C++"
#endif
```