G++ 2.95 for Solaris，\g003\sgi-stl-of-gcc295-for-solaris**memory 完整列表**

```
/*
 * Copyright (c) 1997
 * Silicon Graphics Computer Systems, Inc.
 *
 * Permission to use, copy, modify, distribute and sell this software
 * and its documentation for any purpose is hereby granted without fee,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.  Silicon Graphics makes no
 * representations about the suitability of this software for any
 * purpose.  It is provided "as is" without express or implied warranty.
 *
 */

#ifndef __SGI_STL_MEMORY
#define __SGI_STL_MEMORY

#include <stl_algobase.h>
#include <stl_alloc.h>
#include <stl_construct.h>
#include <stl_tempbuf.h>
#include <stl_uninitialized.h>
#include <stl_raw_storage_iter.h>


#if defined(__STL_MEMBER_TEMPLATES)

__STL_BEGIN_NAMESPACE

template <class _Tp>
class auto_ptr {
private:
  _Tp* _M_ptr;         // 侯捷：no owns ?

public:
  typedef _Tp element_type;
  explicit auto_ptr(_Tp* __p = 0) __STL_NOTHROW : _M_ptr(__p) {}
  auto_ptr(auto_ptr& __a) __STL_NOTHROW : _M_ptr(__a.release()) {}
  template <class _Tp1> auto_ptr(auto_ptr<_Tp1>& __a) __STL_NOTHROW
    : _M_ptr(__a.release()) {}
  auto_ptr& operator=(auto_ptr& __a) __STL_NOTHROW {
    if (&__a != this) {
      delete _M_ptr;
      _M_ptr = __a.release();
    }
    return *this;
  }
  template <class _Tp1>
```

```
  auto_ptr& operator=(auto_ptr<_Tp1>& __a) __STL_NOTHROW {
    if (__a.get() != this->get()) {
      delete _M_ptr;
      _M_ptr = __a.release();
    }
    return *this;
  }
  ~auto_ptr() __STL_NOTHROW { delete _M_ptr; }

  _Tp& operator*() const __STL_NOTHROW {
    return *_M_ptr;
  }
  _Tp* operator->() const __STL_NOTHROW {
    return _M_ptr;
  }
  _Tp* get() const __STL_NOTHROW {
    return _M_ptr;
  }
  _Tp* release() __STL_NOTHROW {
    _Tp* __tmp = _M_ptr;
    _M_ptr = 0;
    return __tmp;
  }
  void reset(_Tp* __p = 0) __STL_NOTHROW {
    delete _M_ptr;
    _M_ptr = __p;
  }

  // According to the C++ standard, these conversions are required.  Most
  // present-day compilers, however, do not enforce that requirement---and,
  // in fact, most present-day compilers do not support the language
  // features that these conversions rely on.

#ifdef __SGI_STL_USE_AUTO_PTR_CONVERSIONS

private:
  template<class _Tp1> struct auto_ptr_ref {
    _Tp1* _M_ptr;
    auto_ptr_ref(_Tp1* __p) : _M_ptr(__p) {}
  };

public:
  auto_ptr(auto_ptr_ref<_Tp> __ref) __STL_NOTHROW
    : _M_ptr(__ref._M_ptr) {}
  template <class _Tp1> operator auto_ptr_ref<_Tp1>() __STL_NOTHROW
    { return auto_ptr_ref<_Tp>(this->release()); }
  template <class _Tp1> operator auto_ptr<_Tp1>() __STL_NOTHROW
    { return auto_ptr<_Tp1>(this->release()); }
```

```
#endif /* __SGI_STL_USE_AUTO_PTR_CONVERSIONS */
};

__STL_END_NAMESPACE
#endif /* member templates */

#endif /* __SGI_STL_MEMORY */


// Local Variables:
// mode:C++
// End:
```