

# HP aC++ version A.05.36 and HP ANSI C version A.05.36 Compiler Products

5187-0137

June 2002

Copyright 2002 Hewlett-Packard Company

## Legal Notices

Copyright (C) Hewlett-Packard Company 2002. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Hewlett-Packard shall not be liable for errors contained herein nor direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Information in this publication is subject to change without notice.

Corporate Offices: Hewlett-Packard Co., 3000 Hanover St., Palo Alto, CA 94304

Use, duplication or disclosure by the U.S. Government Department of Defense is subject to restrictions as set forth in paragraph (b)(3)(ii) of the Rights in Technical Data and Software clause in FAR 52.227-7013.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Use of this document and flexible disc(s), compact disc(s), or tape cartridge(s) supplied for this pack is restricted to this product only.

Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

(C) Copyright 1980, 1984, 1986 AT&T Technologies, Inc. UNIX and System V are registered trademarks of AT&T in the USA and other countries. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

PostScript is a trademark of Adobe Systems, Inc.

(C) Copyright 1985-1986, 1988 Massachusetts Institute of Technology.

X Window System is a trademark of the Massachusetts Institute of Technology.

---

## Preface

This document provides the following information:

- features
- installation information
- related documentation
- problem descriptions and fixes and known limitations

Note: The software code printed in the release notes title indicates the software product version at the time of release. Some product and operating system changes do not require changes to documentation; therefore, do not expect a one-to-one correspondence between these changes and release notes updates.

Unless describing a C++ specific feature or capability, the material provided here covers the C++ as well as ANSI C compilers.

This document resides online in the following locations:

For the aC++ compiler:

```
/opt/aCC/newconfig/RelNotes/ACXX.release.notes
```

For the ANSI C compiler:

```
/opt/ansic/newconfig/RelNotes/ACXX.release.notes
```

To print this file, use an lp command like the following:

For the aC++ compiler:

```
lp -dprinter_name /opt/aCC/newconfig/RelNotes/ACXX.release.notes
```

For the ANSI C compiler:

```
lp -dprinter_name /opt/ansic/newconfig/RelNotes/ACXX.release.notes
```

For the current version of the compilers and their documentation, use your browser to access the HP aC++ compiler and ANSI C compiler home pages.

For the aC++ compiler:

<http://www.hp.com/go/cpp>

For the ANSI C compiler:

<http://www.hp.com/go/c>

## **Problem Reporting:**

If you have any problems with the software or documentation, please contact your local Hewlett-Packard Sales Office or Customer Service Center.

---

# **Chapter 1: Features**

The HP aC++ compiler of this IPF release combines aC++ and C compiler in one. It provides a common ANSI C/C++ compiler with a very high level of compatibility with the previous release of

the HP C compiler. The main goal of this release is to actively drive the migration of ANSI C based HP-UX S/W base from HP C to aC++ compiler with ease.

The new HP C compiler has a very few incompatibilities with its previous release version. These incompatibilities have minimum or no impact on the applications that are migrated to the IPF platform.

The differences are:

- no support for "-Ac" (K&R) mode
- no support for MPE long pointers (^)
- no support for the standalone C tools:  
(lint, cb, cflow, cxref, endif, protogen)
- no support for the HP\_ALIGN pragma (MPE alignments)

This section summarizes the features included in this version of the HP aC++/ HP ANSI C Itanium Processor Family (IPF) compiler. The aC++ compiler supports much of the ISO/IEC 14882 Standard for the C++ Programming Language (the international standard for C++). The ANSI C compiler supports ANSI ISO/IEC 9899:1990.

HP aC++/HP ANSI C provides a variety of performance related options, in addition to the options described in this release notes. See the "HP aC++ Online Programmer's Guide" "Performance" section for full documentation. Chapter 3 of this release notes provides access instructions to the guide.

See "Command Line Options and Pragas" section in the HP aC++ Online Programmer's Guide for more information.

## New Features

New features in HP aC++ version A.05.36 are listed below.

- UTF-16 character support
- OpenMP Standard support
- Standard C++ Library 2.0 based on the new Rogue Wave SL 2.0
- Support for SDK/XDK (cross-compilation)
- aCC\_MAXERR to control maximum number of compiler errors
- Initialized Thread Local Storage
- +O[no]inline=list
- -I- option enhanced to perform prefixinclude search
- +legacy\_hpc and +legacy\_cpp options
- Unified support for packng and alignment pragmas

### UTF-16 character support

This release has a limited support for UTF-16 characters. UTF-16 is described in the Unicode Standard, version 3.0 [UNICODE]. The definitive reference is Annex Q of ISO/IEC 10646-1 [ISO-10646].

The current compiler supports only ASCII strings or characters (8 bit chars with no transliteration) as UTF-16.

Any string or character which is preceded by 'u' is recognized as a UTF-16 literal or character and is stored as an unsigned short type.

Example:

```
#define _UTF16(x) u##x
#define UTF16(y) _UTF16(#y)
typedef unsigned short utf16_t;
utf16_t *utf16_str = UTF16(y); // u"y"
int size = sizeof(u't'); // size of 2 bytes
```

## OpenMP Standard support

This release introduces full support for version 1.0 of the "OpenMP C and C++ Application Program Interface". This specification is available at <http://www.openmp.org/specs>.

To enable recognition of OpenMP pragmas, use the "+Oopenmp" command line option when invoking aCC. This option is effective at any optimization level.

Note: Currently +Onoparallel does not affect the OpenMP pragmas in the source but still disables +Oautopar.

OpenMP programs require the libomp and libcps runtime support libraries to be present on both the compilation and runtime system(s). The compiler driver will automatically include them when linking.

It is recommended that you use the -N option when linking OpenMP programs to avoid exhausting memory when running with large numbers of threads.

For this first release of aCC containing OpenMP, some debugging position information for OpenMP constructs may not be accurate. In addition, symbols marked with the "threadprivate" pragma may not be visible to the debugger. To work around this limitation, use the "\_\_thread" storage class specifier in the symbol declaration instead.

```
#if defined(__HP_aCC) && !defined(__THREAD)
# define __THREAD
#endif

__THREAD int tprvt;
#pragma omp threadprivate(tprvt)
```

along with the command line:

```
aCC -D__THREAD=__thread
```

OpenMP is also supported in HP C and aC++'s ANSI C mode (-Ae).

## Support for SDK/XDK

The SDK/XDK feature helps in selecting components, headerfiles, and libraries installed in alternate locations. You must set either one or both of the following environment variables:

```
SDKROOT
TARGETROOT
```

The SDKROOT environment variable is used as a prefix for all references to tool set components and must be set when you use a non-native development kit or a toolset installed at an alternative location. Some of the toolset

components are compiler drivers, Compiler Applications, Preprocessor, Linker, and object file tools.

Example:

If a compiler tool set is installed in directory `/opt/xdk-ia/` then,

```
export SDKROOT=/opt/xdk-ia
```

prefixes all references to the compiler tool set components with `/opt/xdk-ia`. The following details the default tool set components location as specified in the above command and its earlier location before the execution of the command:

<b>Native Location</b>	<b>Alternate tool set location</b>
<code>/opt/aCC/bin/aCC</code>	<code>/opt/xdk-ia/opt/aCC/bin/aCC</code>
<code>/opt/aCC/lbin/ctcom</code>	<code>/opt/xdk-ia/opt/aCC/lbin/ctcom</code>
<code>/opt/langtools/lbin/ucomp</code>	<code>/opt/xdk-ia/opt/langtools/lbin/ucomp</code>

Invoking the compiler driver 'aCC' results in the invocation of tool set components from the alternate location above.

The `TARGETROOT` environment variable is used as a prefix for all references to target set components and must also be set when using a non-native development kit. Some of the target set components are header files, archive libraries, and shared libraries.

Example:

If a target tool set is installed in directory `/opt/xdk-ia/` then,

```
export TARGETROOT=/opt/xdk-ia
```

prefixes all references to the target tool set components with `/opt/xdk-ia`.

<b>Native Location</b>	<b>Alternate tool set location</b>
<code>/usr/include</code>	<code>/opt/xdk-ia/usr/include</code>
<code>/opt/aCC/include*</code>	<code>/opt/xdk-ia/opt/aCC/include*</code>
<code>/usr/lib</code>	<code>/opt/xdk-ia/usr/lib</code>
<code>/opt/aCC/lib</code>	<code>/opt/xdk-ia/opt/aCC/lib</code>

Environment variables like `LPATH` and options like `-l` or `-L` override `$TARGETROOT` prefixing.

If the compiler is non-native and installed in a different place, the directory path can be prefixed for all references to the compiler.

You have to set the environment variables `XDKROOT` or `SDKROOT` to point to that directory.

For example:

If the compiler is installed in directory `/user/foo`, you can say,

```
export SDKROOT=/user/foo
```

This prefixes all references to the compiler with `/user/foo`.

<b>Default path</b>	<b>New path for compiler</b>
<code>/opt/aCC/bin/aCC</code>	<code>/user/foo/opt/aCC/bin/aCC</code>
<code>/opt/aCC/lbin/ctcom</code>	<code>/user/foo/opt/aCC/lbin/ctcom</code>

```
/opt/langtools/lbin/ucomp /user/foo/opt/langtools/lbin/ucomp
```

For information on SDK usage scenarios, please refer to the paper, "HP-UX Software Development Kit User's Guide" on <http://www.hp.com/dspp>.

### **aCC\_MAXERR to control maximum number of compiler errors**

The aCC\_MAXERR environment variable allows you to set the maximum number of errors you want the compiler to report before it terminates compilation. The current default is 12, but you can set it to any number greater than zero.

The compiler may not be able to recover from all errors and still display:

```
445 Cannot recover from earlier errors
```

instead of

```
699 Error limit reached: halting compilation
```

Example:

The following increases the maximum to 100 errors.

```
$export aCC_MAXERR=100
$aCC -c buggy.c
```

### **Initialized Thread Local Storage**

Static linktime initialization of thread private variables (PODs only) is now supported. Earlier versions of the compiler supported only uninitialized thread private variables.

For example:

```
__thread int j = 2; // allowed with this release
int main()
    j = 20;
}
```

Since thread private memory is allocated during runtime, virtual addresses of the thread private variables should not be used in situations where compile time evaluation of the addresses is necessary. Following are some of the sample incorrect usages:

Ex. #1:

```
__thread int tpv_1;
__thread int *ptr = &tpv_1; //incorrect
```

Ex. #2:

```
__thread int tpv_1;
int *ptr = &tpv_1; //incorrect
```

### **+O[no]inline=list**

The list form is now available. It can contain the names of extern "C" functions or they must be mangled names.

### **-I- option enhanced to perform prefixinclude search**

The `-I` option has been enhanced to do a `prefixinclude` search. The `-I` option, by itself, is not sufficient to handle the case involving a quoted include from a parent file which is not directly on the quoted or bracketed search paths. `Prefixinclude` search provides additional support for the case where, due to use of directory prefixes in `#include` directives in parent including files, the "directory of the including file" is no longer directly on the include search list.

In the non `-I` case, use of directory prefixes in parent `#include` directives causes the compiler to look in some directory offset from the directory of the top-level source file. Analogously, in the `-I` case, use of directory prefixes in parent include files in effect define an offset relative to the directories on the search list. This is equivalent to explicitly specifying the directory prefix explicitly in the child `#include "..."` directive. In fact, modifying the source `#include` directive in this way would allow the intended included file to be found without requiring `prefixinclude` support in the preprocessor.

Here's an example of the problem:

```
$ ls
a.c incl/ mk

$ ls incl
f.h x.h y.h

$ cat a.c
#include "incl/f.h"

$ cat incl/f.h
#include "incl/y.h"
#include "x.h"

$ cat incl/x.h
int x;

$ cat incl/y.h
int y;

$ aCC -c -I. a.c
$ # previous versions of aC++
$ aCC -c -I. -I- -I. a.c
"./incl/f.h", line 2: Error: Could not open include file "x.h".
```

Note that `a.c` compiles fine with `-I.` but with `-I. -I- -I.` it fails to find `x.h` in `-I.`

With the `prefixinclude` feature in effect, the subdirectory prefix (in this case `"incl"`) is inherited from the including file for `#include "..."` style includes. So, if an including file was included as `"prefix/includer"` or `<prefix/includer>` then a file `"includee"` included by `"prefix/includer"` is first searched for using `"prefix/includee"`, and if that fails, is next searched for using `"includee"`. Using each of appropriate `-I` paths.

Searches for `#include <...>` files are not affected by `prefixinclude`, only `#include "..."` file searches have been enhanced.

### **+legacy\_hpc and +legacy\_cpp options**

The new `+legacy_hpc` option in `aCC` can be used to invoke old HP C compiler in "C-mode". The new `+legacy_cpp` option in `aCC` can be used to preprocess the file using an external C preprocessor.

### **Unified Support for Packing and Alignment Pragmas**

The following changes in pragmas were made in this release of aC++ compiler.

- pragma pack - behavior changed
- pragma HP\_ALIGN - is not supported
- pragma align and pragma unalign - are added

For more details see the documentation for A.05.36 under:  
[http://docs.hp.com/hpux/dev/index.html#aC++%20\(ANSI\)](http://docs.hp.com/hpux/dev/index.html#aC++%20(ANSI))

## **New Features in A.05.30 release**

The following features were supported in HP aC++ version A.05.30:

### **HP Caliper is bundled with HP aC++**

HP Caliper 1.0 is a new general-purpose performance analysis and performance improvement tool for IPF applications bundled with HP ANSI C, HP aC++, and HP Fortran 9x. HP Caliper helps you analyze and improve the performance of your native IPF based programs in three ways:

- A simple and quick way to optimize the performance of your program by providing information for compiler profile-based optimization.
- Commands to measure the overall performance of your program.
- Commands to drill down to identify performance parameters of specific functions in your program. A significant advantage of HP Caliper is that it requires no special compilation, link options, or libraries for your IPF based programs.

HP Caliper dynamically measures performance on:

- C, C++, and Fortran 9x binaries
- 32- or 64-bit binaries
- Shared or archive libraries
- Debug or optimized programs

Another advantage is that HP Caliper performs its measurements with low overhead and for well-behaved executables, HP Caliper does not alter the semantic behavior of the target program.

Required disk space is 12 MB. Memory requirements vary with settings used for performance measurement.

### **Documentation**

HP Caliper 1.0 includes the following documentation:

- On-line User Guide in HTML format available with the +help option.
- Printable User Guide in PDF format.
- Command-line option summary available with --help option.
- The caliper (1) man page.

### **Standard C++ Library 2.0 based on the new Rogue Wave SL 2.0**

The -AA command line option enables use of the new 2.0 Standard C++ Library, which includes the new standard conforming ("templated") iostream library. It conforms to the ISO C++ standard.

The 2.0 library is not compatible with the version 1.2.1 Standard C++ Library previously bundled with HP aC++. HP aC++ will continue support for standard C++ library 1.2.1 without name or location change. Customers should not notice any change when -AA is not used. However, the 1.2.1 library is deprecated and will be replaced by the new library eventually.

If you wish to use the new 2.0 library, you must use the -AA option consistently to compile and link all translation units. Mixing object files within an executable is not supported.

The default is for -AA to be on with -Wc,-ansi\_for\_scope,off. The new -AP option turns -AA mode off.

---

## Chapter 2: Installation Information

Read this entire document and any other release notes or readme files you may have before you begin an installation.

To install your software, run the SD-UX swinstall command. It will invoke a user interface that will lead you through the installation. For more information about installation procedures and related issues, refer to "Managing HP-UX Software with SD-UX" and other README, installation, and upgrade documentation provided or described in your HP-UX 11.x operating system package.

Depending on your environment, you may also need documentation for other parts of your system, such as networking, system security, and windowing.

HP aC++ requires approximately 203 MB for the files in  
/opt/aCC

HP ANSI C requires approximately 167 MB for the files in  
/opt/ansic & /opt/ansic/legacy\_hpc

The other components require approximately:  
17 MB for WDB  
63 MB for u2comp.

For more precise sizes, use the command:

```
/usr/sbin/swlist -a size "YourProductNumber"
```

---

## Chapter 3: Related Documentation

Documentation for HP aC++ is described in the following sections.

## Online Documentation

The following online documentation is included with the HP aC++ product:

- "HP aC++ Online Programmer's Guide"

Access the guide in any of the following ways:

- Use the +help command-line option: `/opt/aCC/bin/aCC +help`
- Use the +help command-line option: `/opt/ansic/bin/cc +help`
- From your web browser, enter the appropriate URL:

`file:/opt/aCC/html/C/guide/index.htm` or  
`file:/opt/ansic/html/C/index.htm`

NOTE: All of the files composing the guide are installed in the `/opt/aCC/html/C/` directory. If you choose to move the entire English guide to a different location without having to edit any links, you will need to move all of the subdirectories in `/opt/aCC/html/C/`.

- The guide (excluding Rogue Wave documentation) is also available on the World Wide Web at the following URL: `http://docs.hp.com/hpux/dev/index.html`

- "Using Templates in HP aC++"

This technical document summarizes template features defined in the proposed C++ standard and describes template instantiation as implemented in HP aC++. It is provided with HP aC++ in both postscript and HTML format in the following locations:

`/opt/aCC/newconfig/TechDocs/templates.ps`  
`/opt/aCC/html/C/templates/templates.htm`

- "HP-UX 64-bit Porting and Transition Guide"

Helps developers transition applications from an HP-UX 32-bit platform to the HP-UX 64-bit platform.

Also available on the HP-UX 11.x CD-ROM and on the World Wide Web at the following URL: `http://docs.hp.com/hpux/dev/index.html`

- "HP Linker and Libraries Online User Guide"

To access, use the command: `/usr/ccs/bin/ld +help`

- HP Wildebeest Debugger (HP WDB)

All of the HP WDB documentation is available online in the following directory:

`/opt/langtools/wdb/doc`

The most current HP WDB and its related documentation is available online at the following World Wide Web directory: `http://www.hp.com/go/wdb`

- "Rogue Wave Software Standard C++ Library 2.2.1 Class Reference"

This reference contains an alphabetical listing of all of the classes, algorithms, and function objects in the updated Rogue Wave Standard C++ Library. The library includes the standard iostream library and has namespace `std` enabled.

The reference is provided as HTML formatted files. You can view these files with an HTML browser by opening the following file: `/opt/aCC/html/libstd_v2/stdref/index.htm`  
Or select from the initial window of the "HP aC++ Online Programmer's Guide."

- "Rogue Wave Software Standard C++ Library 2.2.1 User's Guide"

This guide gives information about library usage and includes an extensive discussion of locales and iostreams. The guide is provided as HTML formatted files. You can view these files with an HTML browser by opening the following file:

/opt/aCC/html/libstd\_v2/stdug/index.htm Or select from the initial window of the "HP aC++ Online Programmer's Guide."

- "Rogue Wave Software Standard C++ Library 1.2.1 Class Reference"  
This reference provides an alphabetical listing of all of the classes, algorithms, and function objects in the prior Rogue Wave implementation of the Standard C++ Library. It is provided as HTML formatted files. You can view these files with an HTML browser by opening the following file: /opt/aCC/html/libstd/ref.htm
- "Rogue Wave Software Tools.h++ 7.0.6 Class Reference"  
This reference describes all of the classes and functions in the Tools.h++ Library. It is intended for use with Rogue Wave Standard C++ Library 1.2.1. The reference is provided as HTML formatted files. You can view these files with an HTML browser by opening the following file: /opt/aCC/html/librwtool/ref.htm

NOTE: Although the documentation for Appendix A is supplied with aC++, this is an obsolete interface to Tools.h++ that predates the version that is integrated with the STL. (If building from the Rogue Wave source, you must compile with -DRW\_NO\_STL.) There are 8 templates documented in the main part of the manual as "still supported". This is incorrect. The interfaces for the following 8 templates must be translated to the new interface with two extra template arguments:

RWTPtrHashDictionary	==>	RWTPtrHashMap
RWTPtrHashDictionaryIterator	==>	RWTPtrHashMapIterator
RWTPtrHashTable	==>	RWTPtrHashMultiSet
RWTPtrHashTableIterator	==>	RWTPtrHashMultiSetIterator
RWTValHashDictionary	==>	RWTValHashMap
RWTValHashDictionaryIterator	==>	RWTValHashMapIterator
RWTValHashTable	==>	RWTValHashMultiSet
RWTValHashTableIterator	==>	RWTValHashMultiSetIterator

Refer to defect CR JAGaa90638.

NOTE: Refer to the "HP aC++ Online Programmer's Guide" Information Map for how to obtain additional Rogue Wave documentation and information.

- "HP aC++ Release Notes" is this document. The online ASCII file can be found in /opt/aCC/newconfig/RelNotes/ACXX.release.notes.
- Online manual pages for aCC and c++filt are at /opt/aCC/share/man/man1.Z. Manual pages for the Standard C++ Library and the cfront compatibility libraries (IOStream and Standard Components) are provided under /opt/aCC/share/man/man3.Z.

## Online C++ Example Source Files

Online C++ example source files are located in the directory, /opt/aCC/contrib/Examples/RogueWave. These include examples for the Standard C++ Library and for the Tools.h++ Library.

## Printed Documentation

"HP aC++ Release Notes" is this document. A printed copy of the release notes is provided with the HP aC++ product. Release notes are also provided online, as noted above.

## Other Documentation

Refer to the "HP aC++ Online Programmer's Guide" Information Map for documentation listings, URL's, and course information related to the C++ language. Also, see below.

The following documentation is available for use with HP aC++. To order printed versions of Hewlett-Packard documents, refer to manuals(5).

- "Parallel Programming Guide for HP-UX Systems" (B6056-90006) describes efficient parallel programming techniques available for the HP Fortran 90, HP C, and HP aC++ compilers on HP-UX.

This document is available on the HP-UX 11.x CD-ROM and on the World Wide Web at the following URL: <http://docs.hp.com/hpux/dev/index.html>

To order a paper copy, contact Hewlett-Packard's Support Materials Organization (SMO) at 1-800-227-8164 and provide the above part number.

## HP aC++ World Wide Web Homepage

Access the HP aC++ World Wide Web Homepage at the following URL: <http://www.hp.com/go/cpp>

Refer to the Homepage for the latest information regarding:

- Frequently Asked Questions
- Release Version and Patch Table
- Purchase and Support Information
- Documentation Links
- Compatibility between Releases

## Compatibility between HP aC++ Releases

Maintaining binary compatibility is a key release requirement for new versions of HP aC++. The compiler has maintained the same object model and calling convention and remains compatible with the HP-UX runtime in the code that it generates as well as its intrinsic runtime library (libCsup) across the various releases of HP aC++ and its run-time patch stream.

A.05.36 contains two types of incompatibility with A.05.32:

1. long double is mangled differently.
2. If functions or templates contain more than 10 params, the mangling may be incompatible and unsats may occur.

For the Standard Template Library (libstd) and a generic component/tool library (librwtool), HP aC++ (as well as some other C++ compilers) relies on Rogue Wave's Standard Library and Tools.h++ libraries. From the initial release of HP aC++ through the patch release of version A.01.06, Rogue Wave's Standard Library version 1.2 and Tools.h++ version 7.0.3 compatible libraries were bundled with the compiler.

At the HP aC++ A.01.07 release, the runtime libraries were updated to Rogue Wave's Standard Library version 1.2.1 and Tools.h++ version 7.0.6. These new libraries introduced additional data

members in some base classes resulting in incompatibility with the previous versions.

For more details, refer to the HP aC++ World Wide Web Homepage at the following URL and choose "Compatibility between Releases:" <http://www.hp.com/go/cpp>

### **Difference in Class Size when Compiling in 32-bit versus 64-bit Mode**

The size of a class containing any virtual functions varies when compiled in 32-bit mode versus 64-bit mode. The difference in size is caused by the virtual table pointer (a pointer to an internal compiler table) in the class object. (The pointer is created for any class containing one or more virtual functions.)

When compiling the following example in 32-bit mode, the output is 8.

In 64-bit mode, the output is 16.

```
extern "C" int printf(const char *,...);

class A {
    int a;
public:

    virtual void foo(); //virtual function foo, part of class A
};

void A::foo() {
    return;
}

int main() {
    printf("%d\n", sizeof(A));
}
```

### **Linker Compatibility Warnings**

Beginning with the HP-UX 10.20 release, the linker generates compatibility warnings. These warnings include HP 9000 architecture issues, as well as linker features that may change over time. Compatibility warnings can be turned off with the `+v[no]compatwarnings` linker option. Also, detailed warnings can be turned on with the `+vallcompatwarnings` linker option.

Link time compatibility warnings include the following:

- Duplicate names found for code and data symbols -- The current linker can create a program that has a code and data symbol with the same name. In a future HP-UX release, the linker will adopt a single name space for all symbols. This means that code and data symbols cannot share the same name. Renaming the conflicting symbols solves this problem.
- Unsatisfied symbols found when linking to archive libraries -- If you specify the `-v` option with the `+vallcompatwarnings` option and link to archive libraries, you may see new warnings.

### **Migrating from HP C++ (cfront) to HP aC++**

The compiler lists Errors, Future Errors and Warnings. Expect to see more warnings, errors and future errors reported in your code, many related to standards based syntax. For more complete information, refer to:

1. "HP aC++ Transition Guide" at the following world wide web URL:  
<http://www.hp.com/go/cpp>

The "HP aC++ Online Programmer's Guide" section "Migrating from HP C++ (cfront) to HP aC++" contains a subset of the information found in the transition guide.

2. For general background information and experience, subscribe to the cxx-dev list server (like a notes group). Send a message to majordomo@cxx.cup.hp.com with the following command in the body of the message: subscribe list-name

The information about subscribing to the cxx-dev list server can be obtained from <http://www.hp.com/go/cpp>.

Available list-names are as follows:

cxx-dev	HP C++ Development Discussion List
cxx-dev-announce	HP C++ Development Announcements
cxx-dev-digest	HP C++ Development Discussion List Digest

cxx-dev-announce is also broadcast to cxx-dev, so there is only a need to subscribe to one of the lists. The digest also includes both cxx-dev and cxx-dev-announce.

For additional help or information about the list server, send a message to majordomo@cxx.cup.hp.com with the following command in the body of the message: help

3. For specific support questions, contact your HP support representative.
4. For generic C++ questions, see documents and URL's listed in the "HP aC++ Online Programmer's Guide, Information Map."

Some migration issues are listed below:

- The overload resolution for operators has been updated to reflect the latest version of the ISO/ANSI C++ standard. You may see additional "ambiguous" function error messages displayed.
- Most frequently reported migration issue: `enum x { x1, };`  
The trailing comma is an error, and aC++ generates Warning 921.
- Changes to temporary creation for rvalues used to initialize return values which are const references now causes:

Error 652: Exact position unknown; near file, line#.

Initialization of the result <some const &> requires creating a temporary, yet the temporary's lifetime ends with the return from the function.

- You can bracket your changes with the macro defined by the ISO/ANSI C++ standard. For example:

```
#if __cplusplus >= 199707L
// HP aC++ Code
#endif // __cplusplus >= 199707L
```

- If you are using directed mode instantiation with the cfront based compiler, an awk script can be used to convert your file to an instantiation file that uses the explicit instantiation syntax. Note that explicit instantiation syntax can be used to instantiate a template and all of its member functions, an individual template function, or a template class's member function. The "HP aC++ Online Programmer's Guide" contains an example script.

- In a template, a name with a parameter-dependent qualifier is not taken to be a type unless it is explicitly declared as one with the typename keyword.

You need to explicitly declare a type or a member function type using the typename keyword when all of the following are true:

- The code is inside a template.
- The name is qualified (i.e., it has a "::" token in it).
- The qualifier (to the left of the "::" token) depends on a template parameter.

For example, the following code includes the typename keyword to declare iterator as a type:

```
#include <list>
template <class Element>
class Foo {
public:
    list<Element> e;
    typedef typename list<Element>::iterator MyIterator;
};
```

For more information, refer to the "HP aC++ Transition Guide" at the following World Wide Web URL:

<http://www.hp.com/go/cpp>

---

## Chapter 4: Problem Descriptions and Fixes and Known Limitations

This chapter summarizes the known problems and limitations of the current version of HP aC++ except as otherwise noted.

See the latest "HP-UX Software Status Bulletin" support document for other known problems.

### Known Problems

Customers on support can use the product number to assist them in finding SSB and SRB reports for HP aC++. The product number you can search for is B3910BA.

To verify the product number and version for your HP aC++ compiler, execute the following HP-UX commands:

```
what /opt/aCC/lbin/ctcom
what /opt/aCC/bin/aCC
```

**Following are known problems and workarounds.**

### Object Files Generated at Optimization Level 4

Object files generated by the compiler at optimization level 4, called intermediate object files, are intended to be temporary files. These object files contain an intermediate representation of the user code in a format that is designed for advanced optimizations. Hewlett-Packard reserves the right to change the format of these files

without notice in any compiler release or patch. Use of intermediate files must be limited to the compiler that created them. For the same reason, intermediate object files should not be included into archived libraries that might be used by different versions of the compiler.

When an incompatible intermediate file is detected, the compiler will issue a message and terminate.

### **Lack of support for Precompiled Headers**

Use of Precompiled Header files and the "automatic precompiled header" (+hdr\_cache) option in conjunction with IPF is not supported in this release.

### **Incompatibilities Between the Standard C++ Library Version 1.2.1 and the Draft Standard**

As the ANSI C++ standard has evolved over time, the Standard C++ Library has not always kept up. Such is the case for the "times" function object in the functional header file. In the standard, "times" has been renamed to "multiplies."

If you want to use "multiplies" in your code, to be compatible with the ISO/ANSI C++ standard, use a conditional compilation flag on the aCC command line. For example, for the following program, compile with the command line:

```
aCC -D__HPACC_USING_MULTIPLIES_IN_FUNCTIONAL test.c

// test.c
int times; //user defined variable
#include <functional>
// multiplies can be used in

int main() {}
// end of test.c
```

This flag and the following flags are now automatically set with IPF:

- -D\_HPACC\_THREAD\_SAFE\_RB\_TREE
- -D\_\_HPACC\_USING\_MULTIPLIES\_IN\_FUNCTIONAL
- -D\_\_HPACC\_FIX\_FUNC\_ADAPTER\_OPERATOR
- -D\_\_HPACC\_FULL\_ITERATOR\_REL\_OPS
- -D\_\_HPACC\_TEMPLATE\_PAIR\_CTOR
- -D\_\_HPACC\_MEM\_FUN\_ADAPTOR

### **Conflict between macros.h and numeric\_limits Class (min and max)**

If your code includes /usr/include/macros.h, note that the min and max macros defined in macros.h conflict with the min and max functions defined in the numeric\_limits class of the Standard C++ Library. The following code, for example, would generate a compiler Error 134:

```
numeric_limits<unsigned int>::max();
```

If you must use the macros.h header, try undefining the macros that conflict:

```
...
#include <macros.h>
#undef max
#undef min
...
```

## Known Limitations

Some of these limitations will be removed in future releases of HP aC++. Please be aware that some of these limitations are platform-specific.

- Limitation when Unloading Shared Libraries in an Application

Normally, at program termination (exit) or at a call to `shl_unload()` or `dlclose()`, all explicitly loaded libraries are closed automatically and static destructors are executed at that time.

When an application calls `shl_unload()` or `dlclose()` and that causes `libcsup` to be unloaded, it fails when it executes static destructors at program termination. This causes a program abort, since related code and data of `libcsup` are no longer present.

See defect JAGaa86491.

- HP aC++ does not support large files (i.e., greater than 2 GB) with `<iostream.h>` or `<iostream>`.
- Known limitations of exception handling features:
  - Interoperability with `setjmp/longjmp` (undefined by the ISO/ANSI C++ international standard) is unimplemented. Executing `longjmp` does not cause any destructors to be run.
  - If an unhandled exception is thrown during program initialization phase (that is, before the main program begins execution) destructors for some constructed objects may not be run.
  - Symbolic debugging information is not always emitted for objects which are not directly referenced. For instance, if a pointer to an object is used but no fields are ever referenced, then HP aC++ only emits symbolic debug information for the pointer type and not for the type of object that the pointer points to. For instance, use of `Widget *` only emits debug information for the pointer type `Widget *` and not for `Widget`. If you wish such information, you can create an extra source file which defines a dummy function that has a parameter of that type (`Widget`) and link it into the executable program.
- Known limitations of signal handling features:
  - Throwing an exception from a signal handler is not supported, since a signal can occur anywhere, including optimized regions of code in which the values of destructible objects are temporarily held in registers. Exception handling depends on destructible objects being up-to-date in memory, but this condition is only guaranteed at call sites.
  - Issuing a `longjmp` in a signal handler is not recommended for the same reason that throwing an exception is not supported. The signal handler interrupts processing of the code resulting in undefined data structures with unpredictable results.
- Source-level debugging of C++ shared libraries is supported. However, there are limitations related to debugging C++ shared libraries, generally associated with classes whose member functions are declared in a shared library, and that have objects declared outside the shared library where the class is defined. Refer to the appropriate release notes and manuals for the operating system and debugger you are using. Refer also to the Software Status Bulletin for additional details.
- Instantiation of shared objects with virtual functions in shared memory is not supported.

- Using `shl_load(3X)` with Library-Level Versioning

Once library-level versioning is used, calls to `shl_load()` (see `shl_load(3X)`) should specify the actual version of the library that is to be loaded. For example, if `libA.so` is now a symbolic link to `libA.so.1`, then calls to dynamically load this library should specify the latest version available when the application is compiled, such as:

```
shl_load("libA.so.1", BIND_DEFERRED, 0);
```

This will insure that, when the application is migrated to a system that has a later version of `libA` available, the actual version desired is the one that is dynamically loaded.

- Memory Allocation Routine `alloca()`

The compiler supports the built in function, `alloca`, defined in the `/usr/include/alloca.h` header file. The implementation of the `alloca()` routine is system dependent, and its use is not encouraged. `alloca()` is a memory allocation routine similar to `malloc()` (see `malloc(3C)`). The syntax is:

```
void *alloca(size_t <size>);
```

`alloca()` allocates space from the stack of the caller for a block of at least `<size>` bytes, but does not initialize the space. The space is automatically freed when the calling routine exits.

Note, memory returned by `alloca()` is not related to memory allocated by other memory allocation functions. Behavior of addresses returned by `alloca()` as parameters to other memory functions is undefined.

To use this function, you must use the `<alloca.h>` header file.