



Getting Started with C for AIX

Introduction and Installation Guide

Note!

Before using this information and the product it supports, be sure to read the information in "Notices" on page 33.

First Edition (June, 2002)

This edition applies to Version 6 Release 0 Modification 0 of IBM C for AIX (product number 5765-F57) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM® welcomes your comments. You can send them by either of the following methods:

- Internet: compinfo@ca.ibm.com

Be sure to include your e-mail address if you want a reply.

- By mail to the following address:

IBM Canada Ltd. Laboratory
Information Development
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. C for AIX Overview 1

Command-Line C Compiler	1
Libraries.	2
IBM Distributed Debugger.	2
Other Tools and Utilities	2
C for AIX Documentation and Online Help	3
Accessing the Online Documentation	3
Accessing Additional Information	3

Chapter 2. New and Special C for AIX Features 5

Efficiency, Optimization, and Special Options	5
New C for AIX Options	5
New C for AIX Pragmas	6
Other New C for AIX Features	6
New C for AIX Built-In Functions	6
Enhanced Language Level Support	8
Conformance to Industry Standards	8
ISO/IEC 9899:1990 International Standard Compatibility	8
ISO/IEC 9899:1999 International Standard Support	8
Program Parallelism and OpenMP Compliance	10
Portability	11
Features Related to GNU C Compilers	11
32-Bit and 64-Bit Application Development	14
Predefined Macros for C99 Features, Features Related to GNU C and Other IBM Extensions	18
National Language Support	20

Chapter 3. Installing C for AIX. 21

System Requirements	21
Prerequisite Tasks and Conditions	22
Installing C for AIX.	23
Installing C for AIX by Selecting Filesets.	23
Installing All the Components of C for AIX.	24
Installing C for AIX Over a Network	24
After Installing C for AIX.	25
Installing C for AIX to a Non-Default Directory	25
Enrolling Licenses with LUM	27
Enrolling Licenses Using the LUM GUI	27
Enrolling LUM Licenses From the Command Line	28
Installing Fixes and Upgrades for C for AIX	28
Uninstalling C for AIX.	29
C for AIX Packaging and Filesets	29
Filesets Required for C Compiler	30
Filesets Required for C Compiler Online Help.	30
Filesets Required for IBM Distributed Debugger	31
Filesets Required to Run LUM and SMIT Client GUIs	31
C++ Runtime Filesets Required to Run C for AIX	31
Optional C for AIX Filesets	32
Upgrading the AIX Operating System	32

Notices 33

Programming Interface Information	35
Trademarks and Service Marks	35
Industry Standards	35

Chapter 1. C for AIX Overview

Read this chapter to find out:

- What components and tools are included in IBM C for AIX®, Version 6.0?
- How can I access and use the online help? Where can I go for additional information?

IBM C for AIX, Version 6.0 is a command-line C compiler for the AIX operating system. Libraries and tools shipped with the compiler help you to effectively create complex programs. The Distributed Debugger shipped with the compiler allows you to visually debug programs running in a client-server environment. Extensive online help ensures that you can easily access the helpful information you need for using C for AIX.

The C for AIX compiler builds on the AIX operating system and the PowerPC® architecture to offer many efficient compiler options and built-in functions. It also allows users to compile industry-standards compliant programs for running on both 32-bit and 64-bit environments. In addition, this new version of the compiler offers:

- New compiler options and pragmas for increased efficiency
- Conformance to industry standards
- Optimizations exploiting the PowerPC architecture
- Support for a subset of features related to GNU C
- Enhanced language level support

For overview information about the components included in C for AIX, see:

- “Command-Line C Compiler”
- “Libraries” on page 2
- “IBM Distributed Debugger” on page 2
- “Other Tools and Utilities” on page 2

For additional information, see:

- “Accessing the Online Documentation” on page 3
- “Accessing Additional Information” on page 3

Command-Line C Compiler

You can use C for AIX as a C compiler for files with a .c (lowercase c) suffix. The compiler processes your text-based program source files to create an executable object module.

In most cases, you should use the `xlC` command to compile C source files.

Compiler options perform a wide variety of functions, such as setting compiler characteristics, describing the object code and compiler output, and performing some preprocessor functions. You can specify compiler options on the command line or in a compiler configuration file. You can also specify a limited number of options in the source file. For more information about new compiler options for C

for AIX, refer to “New C for AIX Options” on page 5. For detailed information about the various compiler options, refer to the *C for AIX Compiler Reference*.

RELATED REFERENCES

- Compiler Command Line Options in *C for AIX Compiler Reference*

Libraries

C for AIX is shipped with the following libraries.

- SMP Runtime Library supports both explicit and automated parallel processing.
- Memory Debug Runtime is used for diagnosing memory leaks.

IBM Distributed Debugger

C for AIX is shipped with IBM Distributed Debugger.

The IBM Distributed Debugger is a client/server application that enables you to detect and diagnose errors in your programs. This client/server design makes it possible to debug both programs running on systems that are accessible through a network connection and programs that are on your workstation.

The debugger server, also known as a *debug engine*, runs on the same system where the program you want to debug runs. This system can be your workstation or a system accessible through a network. If you debug a program running on your workstation, you are performing *local debugging*. If you debug a program running on a system accessible through a network connection, you are performing *remote debugging*.

The Distributed Debugger client is a graphical user interface where you can issue commands used by the debug engine to control the execution of your program. For example, you can set breakpoints, step through your code, and examine the contents of variables. The Distributed Debugger user interface lets you debug multiple applications, which might be written in different languages, from a single debugger session. Each program you debug is shown on a separate program page. The type of information that is displayed depends on the debug engine that you are connected to.

By default, the Distributed Debugger is installed to the `/usr/idebug` directory. To start the Distributed Debugger, type `idebug` on the command line.

Other Tools and Utilities

For more information about these tools, commands and utilities, refer to the *C for AIX Compiler Reference*.

CreateExportList Command

Creates a file containing a list of all the global symbols found in a given set of object files.

RELATED REFERENCES

- CreateExportList Command in *C for AIX Programming Topics and Utilities*

C for AIX Documentation and Online Help

This section provides an overview of the documentation offered with the product. Information about the compiler, its various tools and utilities, and the C and C++ programming languages is provided through online HTML-based help and Portable Document Format (PDF) publications. You can install the help at the same time as you install C for AIX.

A plain-text overview of C for AIX commands and compiler options is also provided. Use your text editor to open and `/usr/lib/nls/LANG/vac.help` file for C compiler help. *LANG* represents the language and location code. For example, `en_US` is the language and location code for US English (ISO) help.

RELATED REFERENCES

- “Accessing the Online Documentation”
- “Accessing Additional Information”
- “Installing C for AIX” on page 23
- “C for AIX Packaging and Filesets” on page 29

Accessing the Online Documentation

C for AIX provides you with extensive online documentation. To view the HTML-format online help, you need to have a frames-capable browser such as Netscape Communicator Version 4.04 (or later) installed on your system.

To access the online help for C for AIX, run the `cforaixhelp` command from the `/usr/vac/bin` directory.

You can also access the online help from your CDE desktop, by selecting **Application Manager > C for AIX > Help Homepage**.

Much of the online documentation is also available in Adobe Portable Document Format (PDF) format. You can view and print this information using the Adobe Acrobat Reader. If you do not already have the Adobe Acrobat Reader program installed, you can download from the Adobe Web site at <http://www.adobe.com>. The Adobe PDF files are in the `/usr/vac/pdf` directory.

The following PDF documents are shipped with C for AIX and provide detailed explanation of a number of topics covered in this document.

- *C for AIX C/C++ Language Reference* (`language.pdf`) contains information about C and C++ programming languages, as used by IBM, including the new portability and standardization extensions.
- *C for AIX Compiler Reference* (`compiler.pdf`) contains information about the various compiler options, pragmas, macros, and built-in functions, including those used for parallel processing.
- *IBM C for AIX, Version 6.0 License Information* (`license.pdf`) contains important information about the product license.
- *IBM Distributed Debugger* (`debug.pdf`) contains information about the IBM Distributed Debugger tool.

Accessing Additional Information

For the latest information about C for AIX, visit the IBM C for AIX Web site at <http://www.ibm.com/software/ad/caix>.

Chapter 2. New and Special C for AIX Features

Read this section to find out about the new IBM C for AIX features, including:

- New compiler options and pragmas for increased efficiency
- Conformance to industry standards
- Optimizations exploiting the PowerPC architecture
- Support for a subset of features related to GNU C
- Enhanced language level support

Efficiency, Optimization, and Special Options

IBM C for AIX, Version 6.0 offers new compiler options, pragmas, built-in functions, and other features for ease of development and efficiency of compilation.

RELATED REFERENCES

- “New C for AIX Options”
- “New C for AIX Pragmas” on page 6
- “Other New C for AIX Features” on page 6
- “New C for AIX Built-In Functions” on page 6
- “New Built-In Functions for PowerPC Processors” on page 6

New C for AIX Options

New and changed compiler options are described in detail in the *C for AIX Compiler Reference*.

- The new `-qhot` option determines whether to perform high-order transformations on loops and array language during optimization and whether to pad array dimensions and data objects to avoid cache misses.
- The new `-qarch=pwr4` suboption supports POWER4 architecture.
- The new `-qtune=pwr4` suboption supports POWER4 architecture.
- The new `-qtocmerge` option enables the TOC merge feature in C for AIX. This feature maps data items to absolute addresses by means of an import file.
- The new `-qreport` option determines whether to produce transformation reports showing how the program is parallelized and how loops are optimized.
- The new `-qipa=threads[=N]` suboption permits the running of a number of parallel threads.
- `-qlargepage` is a new option to support the new POWER4 architecture’s support of large 16M pages, in addition to the default 4K pages.
- The new `-qsmallstack` optimization option is intended to reduce the size of the stack frame.
- The new `-qunwind` optimization option allows the compiler to reduce the number of times that registers are saved and restored on the most common paths through a function, if no exception will be thrown through a function.

Unsupported Compiler Options

The following options are no longer supported in C for AIX.

- `-qusepcomp` and `-qgenpcomp` options for precompiled headers
- `-qonce` option to avoid including a header file more than once

New C for AIX Pragmas

New pragmas are described in detail in the *C for AIX Compiler Reference*.

- The new `#pragma execution_frequency(very_low)` directive specifies infrequently-executed code.
- The new `#pragma snapshot` directive sets a debugging breakpoint at the point of the pragma, and defines a list of variables to examine when program execution reaches that point.
- The new `#pragma pack` directive allows the users to modify the alignment rule for members of structures.

Other New C for AIX Features

- The maximum bit-field length is increased to 64 bits.

RELATED REFERENCES

- Declaring and Using Bit Fields in Structures in *C for AIX C/C++ Language Reference*

New C for AIX Built-In Functions

Several new built-in functions are supported by PowerPC processors. They are listed in “New Built-In Functions for PowerPC Processors” and described in detail in the *C for AIX Compiler Reference*. In addition, the following new built-in functions were added for C for AIX:

- The `__prefetch_by_stream` built-in function enables you to load specific data from main memory into the cache.
- The `__iospace_lwsync` and `__isync` built-in function can be used to specify the type of synchronization, and where and how it should take place.

New Built-In Functions for PowerPC Processors

The following new built-in functions are supported by PowerPC processors.

New built-in functions for PowerPC processors

Function	Description
<code>__clear_lock_up</code>	Clear Lock on UniProcessor systems
<code>__clear_lock_mp</code>	Clear Lock on MultiProcessor systems.
<code>__check_lock_up</code>	Check Lock on UniProcessor systems.
<code>__check_lock_mp</code>	Check Lock on MultiProcessor systems.
<code>__clear_lockd_up</code>	Clear Lock Doubleword on UniProcessor systems.*
<code>__clear_lockd_mp</code>	Clear Lock Doubleword on MultiProcessor systems.*
<code>__check_lockd_up</code>	Check Lock Doubleword on UniProcessor systems.*
<code>__check_lockd_mp</code>	Check Lock Doubleword on MultiProcessor systems.*
<code>__rdlam</code>	Rotate Double Left and AND with Mask.*
<code>__rldimi</code>	Rotate Left Doubleword Immediate then Mask Insert.*
<code>__rlwimi</code>	Rotate Left Word Immediate then Mask Insert.
<code>__rlwnm</code>	Rotate Left Word then AND with Mask.
<code>__rotatel8</code>	Rotate Left Doubleword.*
<code>__rotatel4</code>	Rotate Left Word.
<code>__fctid</code>	Floating Convert to Integer Doubleword.*

New built-in functions for PowerPC processors

Function	Description
<code>__fctidz</code>	Floating Convert to Integer Doubleword with Rounding towards Zero. *
<code>__fctiw</code>	Floating Convert To Integer Word.
<code>__fctiwz</code>	Floating Convert To Integer Word with Rounding towards Zero.
<code>__fcfid</code>	Floating Convert From Integer Doubleword. *
<code>__stfiw</code>	Store Floating-Point as Integer Word.
<code>__mtfsfi</code>	Move to FPSCR Field Immediate.
<code>__mtfsf</code>	Move to FPSCR Fields.
<code>__mtfsb0</code>	Move to FPSCR Bit 0.
<code>__mtfsb1</code>	Move to FPSCR Bit 1.
<code>__mulhw</code>	Multiply High Word Signed.
<code>__mulhwu</code>	Multiply High Word Unsigned.
<code>__mulhd</code>	Multiply High Doubleword Signed.*
<code>__mulhdu</code>	Multiply High Doubleword Unsigned*
<code>__tw</code>	Trap Word.
<code>__tdw</code>	Trap Doubleword.*
<code>__eieio</code>	Extra name for the existing <code>__iospace_eieio</code> built-in function.
<code>__sync</code>	Extra name for the existing <code>__iospace_sync</code> built-in function.
<code>__lwsync</code>	Extra name for the existing <code>__iospace_lwsync</code> built-in function.
* This built-in function is valid in 64-bit mode only.	

Enhanced Language Level Support

The `-q1anglvl` compiler option is used to specify the supported language level, and therefore affects the way your code is compiled.

For example, to compile your C programs in a way that strictly conforms to the ISO/IEC 9899:1990 International Standard (C89), you need to specify `-q1anglvl=stdc89`. The `stdc89` suboption instructs the compiler to strictly enforce the standard, and not to allow any language extensions.

You can also use extensions to the standard language levels. Extensions that do not interfere with the standard features are called *orthogonal* extensions. For example, when you compile C programs, you can enable extensions that are orthogonal to C89 by specifying `-q1anglvl=extc89`.

Most of the language features described in the ISO/IEC 9899:1999 International Standard (C99) are considered orthogonal extensions to C89.

In general, a valid program that compiles and runs correctly under a standard language level should continue to compile correctly and run to produce the same result with the orthogonal extensions enabled.

Non-orthogonal extensions, on the other hand, can interfere or conflict with aspects of the language as described in one of the international standards. Acceptance of these extensions must be explicitly enabled by individual compiler options.

The main suboptions for the `-qlanglvl` option are listed below.

Important `-qlanglvl` Suboptions

-qlanglvl Suboption	Suboption Description
<code>-qlanglvl=stdc99</code>	Specifies strict conformance to the C99 standard.
<code>-qlanglvl=stdc89</code>	Specifies strict conformance to the C89 standard.
<code>-qlanglvl=ansi</code>	Specifies strict conformance to the C89 standard and enables the <code>-qlonglong</code> compiler option.
<code>-qlanglvl=extc99</code>	Enables all the orthogonal extensions on top of C99.
<code>-qlanglvl=extc89</code>	Enables all the orthogonal extensions on top of C89.
<code>-qlanglvl=extended</code>	Enables all the orthogonal extensions on top of C89 and specifies the <code>-qupconv</code> compiler option.

RELATED REFERENCES

- `langlvl` in *C for AIX Compiler Reference*
- `upconv` in *C for AIX Compiler Reference*
- `longlong` in *C for AIX Compiler Reference*
- The IBM Language Extensions in *C for AIX C/C++ Language Reference*
- The IBM C Language Extensions in *C for AIX C/C++ Language Reference*

Conformance to Industry Standards

IBM C for AIX, Version 6.0 builds on accepted industry standards, so your code can be ported easily.

RELATED REFERENCES

- “ISO/IEC 9899:1990 International Standard Compatibility”
- “ISO/IEC 9899:1999 International Standard Support”
- “Program Parallelism and OpenMP Compliance” on page 10
- “Enhanced Language Level Support” on page 7

ISO/IEC 9899:1990 International Standard Compatibility

The ISO/IEC 9899:1990 International Standard (also known as C89) specifies the form and establishes the interpretation of programs written in the C programming language. This International Standard is designed to promote the portability of C programs among a variety of implementations. This Standard was amended and corrected by ISO/IEC 9899/COR1:1994, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996. To ensure that C for AIX compiles your code in a way that enforces the amended and corrected C89 standard, specify the `-qlanglvl=stdc89` compiler option.

ISO/IEC 9899:1999 International Standard Support

The ISO/IEC 9899:1999 International Standard (C99) is an updated standard for programs written in the C programming language. It is designed to enhance the capability of the C language, provide clarifications and incorporate technical

corrections to C89. C for AIX supports many features of this standard. To ensure that the compiler enforces support for these features, specify the `-q|lang|v1=stdc99` compiler option.

Not all runtime functions and facilities required by the ISO/IEC 9899:1999 International Standard are supported on all the operating system levels that can run this version of the compiler. The availability of system header file provides an indication of the support.

RELATED REFERENCES

- “Major New Features in C99”
- “Changes and Clarifications of C89 Supported in C99” on page 10

Major New Features in C99

The following C99 features are new in C for AIX.

ISO/IEC 9899:1999 International Standard Extensions to IBM C

New C99 Feature	Related Reference
restrict type qualifier for pointers	The restrict Type Qualifier in <i>C for AIX C/C++ Language Reference</i>
universal character names	The Unicode Standard in <i>C for AIX C/C++ Language Reference</i>
predefined identifier <code>__func__</code>	Predefined Identifiers in <i>C for AIX C/C++ Language Reference</i>
function-like macros with variable and empty arguments	Function-Like Macros in <i>C for AIX C/C++ Language Reference</i>
<code>_Pragma</code> unary operator	The <code>_Pragma</code> Operator in <i>C for AIX C/C++ Language Reference</i>
variable length array	Arrays in <i>C for AIX C/C++ Language Reference</i>
static keyword in array index declaration	Arrays in <i>C for AIX C/C++ Language Reference</i>
complex data type	Complex Types in <i>C for AIX C/C++ Language Reference</i>
long long int and unsigned long long int types	Integer Variables in <i>C for AIX C/C++ Language Reference</i>
hexadecimal floating-point constants	Hexadecimal Floating Constants in <i>C for AIX C/C++ Language Reference</i>
compound literals for aggregate types	Compound Literals in <i>C for AIX C/C++ Language Reference</i>
designated initializers	Initializers in <i>C for AIX C/C++ Language Reference</i>
C++ style comments	Comments in <i>C for AIX C/C++ Language Reference</i>
implicit function declaration not permitted	Function Declarations in <i>C for AIX C/C++ Language Reference</i>
mixed declarations and code	for Statement in <i>C for AIX C/C++ Language Reference</i>
<code>_Bool</code> type	Simple Type Specifiers in <i>C for AIX C/C++ Language Reference</i>

New C99 Feature	Related Reference
inline function declarations	Inline Functions in <i>C for AIX C/C++ Language Reference</i>
initializers for aggregates	Initializing Arrays Using Designated Initializers in <i>C for AIX C/C++ Language Reference</i>

Changes and Clarifications of C89 Supported in C99

The following C99 features are major changes and clarifications of the C89 standard.

- Flexible array members are allowed. The last member of a structure with two or more members can be declared without the size.
- Declaring implicit `int` is not supported. All declarations must have a type specifier.
- Multiplicative operators truncate towards zero. Integer division discards the fractional part of the algebraic result.
- Left shift of signed nonnegative integer is now defined.
- Trailing commas are allowed in enumeration specifiers.
- Duplicate type qualifiers are accepted and ignored, unless explicitly specified otherwise.
- A diagnostic message will be issued if a required expression is missing from the return statement.
- Constant expressions evaluated during preprocessing now use `long long` and `unsigned long long` data types.
- Empty macro arguments are allowed in function-like macros.
- The maximum value of `#line` has increased to 2 147 483 647.

RELATED REFERENCES

- Declaring and Defining a Structure in *C for AIX C/C++ Language Reference*
- Type Specifiers in *C for AIX C/C++ Language Reference*
- Division `/` in *C for AIX C/C++ Language Reference*
- Bitwise Left and Right Shift `<<>>` in *C for AIX C/C++ Language Reference*
- Declaring an Enumeration Data Type in *C for AIX C/C++ Language Reference*
- The `const` Type Qualifier in *C for AIX C/C++ Language Reference*
- `return` Statement in *C for AIX C/C++ Language Reference*
- Integer Variables in *C for AIX C/C++ Language Reference*
- Function-Like Macros in *C for AIX C/C++ Language Reference*
- Line Control (`#line`) in *C for AIX C/C++ Language Reference*

Program Parallelism and OpenMP Compliance

The OpenMP Application Program Interface (API) supports multi-platform shared-memory parallel programming in C, C++ and Fortran on all architectures, including UNIX[®] platforms and Windows NT[®] platforms. OpenMP is a portable, scalable model that gives programmers a simple and flexible interface for developing shared-memory parallel applications. It is jointly defined by a group of major computer hardware and software vendors, including IBM.

C for AIX is OpenMP Specification 1.0 compliant, since it recognizes and preserves the semantics of all its elements. The directives, library functions, and environment variables described below allow you to create and manage parallel programs while permitting portability.

C for AIX has had OpenMP Specification 1.0 support since Version 5. You should use this industry standard, rather than the old IBM-specific symmetric multi-processing (SMP) directives.

To enable OpenMP parallel processing, you must specify the `-qsmp` compiler option.

- To choose automated parallelism, specify `-qsmp` or `-qsmp=auto`. This suboption enables the compiler to perform *implicit parallelism*, in addition to recognizing and implementing any OpenMP directives, library functions, and environment variables included in the program.
- To choose strict compliance to the OpenMP Specification 1.0, specify `-qsmp=omp`. This suboption ensures that the compiler implements only the OpenMP directives, library functions, and environment variables specified in the code. It does not perform any additional automated parallel processing.

For details about parallel processing and the C for AIX implementation of the OpenMP Specification 1.0, refer to the *C for AIX Compiler Reference*.

For more information about the OpenMP Specification, visit the OpenMP Web site at <http://www.openmp.org>.

RELATED REFERENCES

- Pragmas to Control Parallel Processing in *C for AIX Compiler Reference*

RELATED CONCEPTS

- Program Parallelization in *C for AIX Compiler Reference*

Portability

Features Related to GNU C Compilers

To ease porting of code written for GNU C, some GNU C features are supported in C for AIX. If you have code written for GNU C, you can use this section to help you port your code to C for AIX.

To use *supported* features with your C code, specify one of `-qlanglvl=extended`, `-qlanglvl=extc89`, or `-qlanglvl=extc99`.

The compiler recognizes *accept/ignore* features as acceptable programming keywords, but does not support them.

Compiling source code that uses these features under a strict `-qlanglvl` suboption will result in error messages.

For more information on GNU C, see <http://www.gnu.org/software/gcc/>.

RELATED REFERENCES

- “GCC Function Attributes” on page 12

- “GCC Variable Attributes”
- “GNU C Type Attributes” on page 13
- “GNU C Assertions” on page 13
- “Other Extensions Related to GNU C” on page 14

GCC Function Attributes

Use the keyword `__attribute__` to specify special attributes when making a function declaration. This keyword is followed by an attribute specification inside double parentheses.

GCC function attribute compatibility with C for AIX

Function Attribute	Behavior
alias	accept/ignore
cdecl	accept/ignore
const	supported
constructor	accept/ignore
destructor	accept/ignore
dllexport	accept/ignore
dllimport	accept/ignore
eightbit_data	accept/ignore
exception	accept/ignore
format	accept/ignore
format_arg	accept/ignore
function_vector	accept/ignore
interrupt	accept/ignore
interrupt_handler	accept/ignore
longcall	accept/ignore
model	accept/ignore
no_check_memory_usage	accept/ignore
no_instrument_function	accept/ignore
noreturn	supported
pure	supported
regparm	accept/ignore
section	accept/ignore
stdcall	accept/ignore
tiny_data	accept/ignore

RELATED REFERENCES

- Function Attributes in *C for AIX C/C++ Language Reference*

GCC Variable Attributes

Use the keyword `__attribute__` to specify special attributes of variables or structure fields. This keyword is followed by an attribute specification inside double parentheses.

GCC variable attribute compatibility with C for AIX

Variable Attribute	Behavior
aligned	supported
mode	supported
model	accept/ignore
nocommon	accept/ignore
packed	supported
section	accept/ignore
transparent_union	accept/ignore
unused	accept/ignore

RELATED REFERENCES

- Variable Attributes in *C for AIX C/C++ Language Reference*

GNU C Type Attributes

Use the keyword `__attribute__` to specify special attributes of struct and union types when you define these types. This keyword is followed by an attribute specification inside double parentheses. While type attributes are not currently supported in C for AIX, some type attributes are accepted and ignored by the compiler.

GCC type attribute compatibility with C for AIX

Type Attribute	Behavior
aligned	accept/ignore
packed	accept/ignore
transparent_union	accept/ignore
unused	accept/ignore

RELATED REFERENCES

- Type Attributes in *C for AIX C/C++ Language Reference*

GNU C Assertions

Use *assertions* to test what sort of computer or system the compiled program will run on. The assertions `#cpu`, `#machine`, and `#system` are predefined. You can also define assertions by with preprocessing directives `#assert` and `#unassert`.

GNU C assertions in C for AIX

GCC Assertions	Behavior
<code>#assert</code>	supported
<code>#unassert</code>	supported
<code>#cpu</code>	supported
<code>#machine</code>	supported
<code>#system</code>	supported possible values are aix and unix

Other Extensions Related to GNU C

The following features related to GNU C are supported under some non-strict `-qlanglvl` suboptions.

- Use directive `#warning` to cause the preprocessor to issue a warning and continue processing.
- Use directive `#include_next` to specify inclusion of the next header file in a directory after the current one.
- Local labels can be declared at start of each lexical block.
- Refer to the type of an expression with the `typeof` keyword.
- Use compound expressions, conditional expressions, and casts as lvalues.
- Use keyword `__alignof__` to inquire about variable alignment, or the alignment usually required by a type.
- Use alternate spelling of these keywords: `__const__`, `__volatile__`, `__signed__`, `__inline__`, and `__typeof__`.

Under some `-qlanglvl` suboptions, C for AIX recognizes the syntax of the following features. However, the semantics of these features is not supported.

- You can define a global register variable, or a local register variable residing in specified registers.
- Write `__extension__` before an expression containing a GNU C extension to avoid receiving a warning.

Many existing extensions to IBM C/C++ are also supported in GNU C.

RELATED REFERENCES

- Preprocessor Warning Directive (`#warning`) in *C for AIX C/C++ Language Reference*
- Specialized File Inclusion in *C for AIX C/C++ Language Reference*
- Locally Declared Labels in *C for AIX C/C++ Language Reference*
- `typeof` Operator in *C for AIX C/C++ Language Reference*
- Lvalues and Rvalues in *C for AIX C/C++ Language Reference*
- Keywords in *C for AIX C/C++ Language Reference*

32–Bit and 64–Bit Application Development

You can use C for AIX to develop both 32-bit and 64-bit applications. This section outlines various portability considerations in moving C programs from 32-bit to 64-bit mode.

RELATED REFERENCES

- “Constants” on page 15
- “Assignment of Long Variables to Integers and Pointers” on page 16
- “Undeclared Functions” on page 16
- “Structure Sizes and Alignments” on page 17
- “Bit Fields” on page 17
- “Miscellaneous Issues” on page 17
- “Interlanguage Calls with Fortran” on page 18

Constants

The limits of constants change. This table shows changed items in the limits.h header file, their hexadecimal value, and decimal equivalent. The equation gives an idea of how to construct these values.

Changes in Limits of Constants

Type	Hexadecimal	Equation	Decimal
signed long min (LONG_MIN)	0x8000000000000000L	$-(2^{63})$	-9,223,372,036,854,775,808
signed long max (LONG_MAX)	0x7FFFFFFFFFFFFFFFL	$2^{63}-1$	+9,223,372,036,854,775,807
unsigned long max (ULONG_MAX)	0xFFFFFFFFFFFFFFFFUL	$2^{64}-1$	+18,446,744,073,709,551,616

In C and C++, type identification of constants follows explicit rules. However, programs that use constants exceeding the limit (relying on a 2's complement representation) will experience unexpected results in the 64-bit mode. This is especially true of hexadecimal constants and unsuffixed constants, which are more likely to be extended into the 64-bit long type.

Problematic behaviors will generally occur at boundary areas such as:

- constant \geq UINT_MAX
- constant $<$ INT_MIN
- constant $>$ INT_MAX

Some examples of undesirable boundary side effects are:

Undesireable Boundary Side Effects Examples

Constant assigned to long	32 bit mode	64 bit mode
-2,147,483,649 (INT_MIN-1)	+2,147,483,647	-2,147,483,649
+2,147,483,648 (INT_MAX+1)	-2,147,483,648	+2,147,483,648
+4,294,496,726 (UINT_MAX+1)	0	+4,294,967,296
0xFFFFFFFF (UINT_MAX)	-1	+4,294,496,295
0x100000000 (UINT_MAX+1)	0	+4,294,967,296
0xFFFFFFFFFFFFFFFF (ULONG_MAX)	-1	-1

Currently, the compiler gives out of range warning messages when attempting to assign a value larger than the designated range into a long type. This warning message may not appear for every case.

When you bit left-shift a 32-bit constant and assign it into a long type, signed values are sign-extended and unsigned values are zero-extended. The examples in the table below show the effects of performing a bit-shift on both 32- and 64-bit constants, using the following code segment:

```
long l=constantL<<1;
```

Constant Value Bit-Shift Examples

Initial Constant Value	Constant Value after Bit-Shift	
	32-bit	64-bit

Constant Value Bit-Shift Examples

0x7FFFFFFFL (INT_MAX)	0xFFFFFFFFE	0xFFFFFFFFE
0x80000000L (INT_MIN)	0	0x100000000
0xFFFFFFFFFL (UINT_MAX)	0xFFFFFFFFE	0x1FFFFFFFFE

Unsuffix constants can lead to type ambiguity that can impact other parts of your program, such as the result of `sizeof` operations. For example, in 32-bit mode the compiler types a number like 4294967295 (UINT_MAX) as an unsigned long. In 64-bit mode, this same number becomes a signed long. To avoid this possibility, explicitly add a suffix to all constants that have the potential of impacting constant assignment or expression evaluation in other parts of your program. The fix for the above case is to write the number as 4294967295U. This forces the compiler to always see that constant as an unsigned int regardless of compiler mode.

Assignment of Long Variables to Integers and Pointers

Using int and long types in expressions and assignments can lead to implicit conversion through promotions and demotions, or explicit conversions through assignments and argument passing. The following should be avoided:

- Using integer and long types interchangeably, leading to truncation of significant digits or unexpected results.
- Passing long arguments to functions expecting type int.
- Exchanging pointers and int types, causing segmentation faults.
- Passing pointers to a function expecting an int type, resulting in truncation.
- Assignment of long types to float, causing possible loss of accuracy.

Assigning a long constant to an integer will cause truncation without warning. For example:

```
int i;
long l=2147483648; /* INT_MAX+1*/
i=l;
```

What will be the value of i? INT_MAX+1 is 2147483647+1 (0x80000000), which becomes INT_MIN when assigned into a signed type. Truncation occurs because the highest bit is treated as a sign bit. The rule here is that there will be a loss of significant digits.

Similar problems occur when passing constants directly to functions, and in functions that return long types. Making explicit use of the L and UL suffix will avoid most, but not all, problems. Alternately, you can avoid accidental conversions by using explicit prototyping. Another good practice is to avoid implicit type conversion by using explicit type casting to change types.

Undeclared Functions

Any function that returns a pointer should be explicitly declared when compiling in 64-bit mode. Otherwise, the compiler will assume the function returns an int and truncates the resulting pointer, even if you were to assign it into a valid pointer.

Code constructs such as the following are valid in 32-bit mode.

```
a=(char *) calloc(25);
```

However, in 64-bit mode, the pointer `a` will be silently truncated. Even type casting will not prevent the truncation: the memory allocated by `calloc()` was already truncated after the return.

The fix in this case would be to include the appropriate header file, which is `stdlib.h`.

Structure Sizes and Alignments

Structures might present porting problems.

The 64-bit specification changes the size, member and structure alignment of all structures that are recompiled in 64-bit mode. Structures with long types and pointers will generally change size and alignment in 64-bit mode. Some structures may not change in size because they happen to fall on an exact 8-byte boundary even in 32-bit mode.

Sharing data structures between 32- and 64-bit processes is no longer possible unless the structure is devoid of pointer and long types. Unions that attempt to share long and int types, or overlay pointers onto int types will now be aligned differently, or be corrupted. In general, all but the simplest structures must be checked for alignment and size dependencies.

The alignment for `-qalign=full`, `power` or `natural` changes for 64-bit mode. Structure members are aligned on their natural boundaries. Long types and pointer types are word-aligned in 32-bit mode, and doubleword aligned in 64-bit mode. Additional spaces could be used for padding members.

The alignment for `-qalign=twobyte` and `-qalign=mac68k` are not supported in 64-bit mode.

Structures are aligned according to the strictest aligned member. This remains unchanged from 32-bit mode. Because of the padding introduced by the member alignment, structure alignment may not be exactly the same as in the 32-bit mode. This is especially important when you have arrays of structures which contain pointer or long types. The member alignment will change, most likely leading to the structure alignment to change to doubleword alignment (if there are no long long types, double types, and long double types).

Bit Fields

Structure bit fields are limited to 32 bits in 32-bit mode, and can be of type signed int, unsigned int or plain int. Bit fields are packed into the current word. Adjacent bit fields that cross a word boundary will start at storage unit. This storage unit is a word in `power` and `full` alignment, halfword in the `mac68k` and `twobyte` alignment, and byte in the `packed` alignment.

In 32-bit mode, non-integer bit fields are tolerated (but not respected) only in the C extended language level. C++ tolerates non-integer bit fields in any language level.

64-bit bit fields are supported in 64-bit mode.

Miscellaneous Issues

- The `sizeof` operator will now return `size_t` which is an unsigned long.
- The length of the integer required to hold the difference between two pointers is `ptrdiff_t`, and is a signed long type.
- Masks will generally lead to different results when compiled in 64-bit mode from their 32-bit mode behavior.

- Many include files have pointers and structures in them, and their inclusion in 64-bit mode will change the size of your data section even if your program does not use structures and pointers explicitly.
- `__int64` is a long type in 64-bit mode, but will look like a long long type in 32-bit mode. `__int64` types can participate in promotion rules and arithmetic conversion when in 64-bit mode. When in 32-bit mode, these types can not participate in the usual arithmetic conversions.
- In 64-bit mode, member values in a structure passed by value to a `va_arg` argument may not be accessed properly if the size of the structure is not a multiple of 8-bytes. This is a known limitation of the operating system.
- In 64-bit extended mode, zero-extension from unsigned int to an unsigned long preserves the bit pattern. For example, zero-extending an unsigned int with value `0xFFFF FFFF` (large negative value) results in an unsigned long with value `0x0000 0000 FFFF FFFF` (large positive value).

Interlanguage Calls with Fortran

A significant number of applications use C, C++, and Fortran together, by calling each other or sharing files. Such applications are among the early candidates for porting to 64-bit platforms for their abilities to solve larger mathematical models. Experience shows that it is easier to modify data sizes and types on the C side than the Fortran side of such applications. The following table lists C and C++ types and the equivalent Fortran types in the different modes.

Equivalent C/C++ and Fortran Data Types

C/C++ type	32-bit	64-bit
int	INTEGER	INTEGER
unsigned int	LOGICAL	LOGICAL
signed long	INTEGER	INTEGER*8
unsigned long	LOGICAL	LOGICAL*8
pointer	INTEGER	INTEGER*8

In 64-bit mode, Fortran has a `POINTER*8` that is 8 bytes in length as compared to `POINTER` which is 4 bytes in length.

Predefined Macros for C99 Features, Features Related to GNU C and Other IBM Extensions

The following macros are defined have the value of 1 if the listed feature is supported under the specified `qlanglvl` suboption. If the feature is not supported, then the macro is undefined. For description of the features, see “ISO/IEC 9899:1999 International Standard Support” on page 8 and “Features Related to GNU C Compilers” on page 11.

All predefined macros are protected.

Predefined Macros for C99 Features, Features Related to GNU C and Other IBM Extensions

Feature	Predefined Macro Name	Supported in -qlanglvl Suboption
flexible array member	<code>__C99_FLEXIBLE_ARRAY_MEMBER</code>	<code>stdc99, extc99</code>
duplicated type qualifier	<code>__C99_DUP_TYPE_QUALIFIER</code>	<code>stdc99, extc99, extc89, extended</code>
new limit for #line	<code>__C99_MAX_LINE_NUMBER</code>	<code>stdc99, extc99, extc89, extended</code>
<code>_Bool</code> type	<code>__C99_BOOL</code>	<code>stdc99, extc99, extc89, extended</code>

Predefined Macros for C99 Features, Features Related to GNU C and Other IBM Extensions

Feature	Predefined Macro Name	Supported in -qlanglvl Suboption
long long type	__C99_LLONG	stdc99, extc99
inline function specifier	__C99_INLINE	stdc99, extc99, extc89, extended
restrict qualifier	__C99_RESTRICT	stdc99, extc99 -qkeyword=restrict
static keyword in array declaration	__C99_STATIC_ARRAY_SIZE	stdc99, extc99, extc89, extended
universal character name	__C99_UCN	stdc99, extc99, extc89, extended
variable length arrays	__C99_VAR_LEN_ARRAY	stdc99, extc99, extc89, extended
__func__ keyword	__C99_FUNC__	stdc99, extc99, extc89, extended
hexadecimal floating constants	__C99_HEX_FLOAT_CONST	stdc99, extc99, extc89, extended
C++ style comments	__C99_CPLUSCMT	stdc99, extc99
compound literals	__C99_COMPOUND_LITERAL	stdc99, extc99, extc89, extended
designated initialization	__C99_DESIGNATED_INITIALIZER	stdc99, extc99, extc89, extended
mixed declaration and code	__C99_MIXED_DECL_AND_CODE	stdc99, extc99, extc89, extended
function-like macros with variable arguments	__C99_MACRO_WITH_VA_ARGS	stdc99, extc99, extc89, extended
standard pragmas	__C99_STD_PRAGMAS	stdc99, extc99, extc89, extended
_Pragma operator	__C99_PRAGMA_OPERATOR	stdc99, extc99, extc89, extended
complex type	__C99_COMPLEX	stdc99, extc99, extc89, extended
type generic macros <tgmath.h>	__C99_TGMATH	stdc99, extc99, extc89, extended
implicit function declaration not supported	__C99_REQUIRE_FUNC_DECL	stdc99
concatenation of wide string and non-wide string	__C99_MIXED_STRING_CONCAT	stdc99, extc99, extc89, extended
subscripting in non-lvalue arrays	__C99_NON_LVALUE_ARRAY_SUB	stdc99, extc99, extc89, extended
non-constant array initializers	__C99_NON_CONST_AGGR_INITIALIZER	stdc99, extc99, extc89, extended
local labels	__IBM_LOCAL_LABEL	extc99, extc89, extended
labels as values	__IBM_LABEL_VALUE	extc99, extc89, extended
typeof	__IBM_TYPEOF__	__typeof__: extc99, extc89, extended; typeof: -qkeyword=typeof
generalized lvalues	__IBM_GENERALIZED_LVALUE	extc99, extc89, extended
function attributes	__IBM_ATTRIBUTES	extc99, extc89, extended
dollar signs in identifiers	__IBM_DOLLAR_IN_ID	extc99, extc89, extended
variable attributes	__IBM_ATTRIBUTES	extc99, extc89, extended
type attributes	__IBM_ATTRIBUTES	extc99, extc89, extended
__alignof__	__IBM_ALIGNOF__	extc99, extc89, extended
inline functions	__IBM_GCC_INLINE__	__inline__: extc99, extc89, extended -qgcc_inline can also control both __inline__ and inline keywords
explicit register variables	__IBM_REGISTER_VARS	extc99, extc89, extended
alternate keywords	__IBM_ALTERNATE_KEYWORDS	extc99, extc89, extended
__extension__	__IBM_EXTENSION_KEYWORD	extc99, extc89, extended
empty macro arguments	__C99_EMPTY_MACRO_ARGUMENTS	stdc99, extc99, extc89, extended
#assert, #unassert, #cpu, #machine, #system	__IBM_PP_PREDICATE	extc99, extc89, extended
#warning	__IBM_PP_WARNING	extc99, extc89, extended
#include_next	__IBM_INCLUDE_NEXT	extc99, extc89, extended

National Language Support

C for AIX has support that enables you to create international applications, including Unicode support, multibyte character support, and bidirectional support.

RELATED REFERENCES

- *The Unicode Standard in C for AIX C/C++ Language Reference*
- *National Languages Support in VisualAge C++ in C for AIX Compiler Reference*

Chapter 3. Installing C for AIX

This section contains information about the prerequisites, requirements, and environmental considerations for product installation.

C for AIX uses License Use Management (LUM) to control the licenses for the product. LUM is a component of the base operating system in AIX Version 4.3 or higher. You do not need to install LUM, but you might need to configure it before installing C for AIX. You may also want to install the latest version of LUM. It is available for download at <http://www.ibm.com/software/is/lum>. It is also provided on the C for AIX CD in the /lum directory.

After installing C for AIX, you need to enroll your license with License Use Management. LUM documentation, including configuration instructions, is available in PDF at <http://www.ibm.com/software/is/lum/library.html>.

You can also find simplified configuration instructions in the README.password file in /usr/vac.

You need to install and configure LUM only once. If LUM is already installed and configured on your system, you can skip to “Installing C for AIX” on page 23.

If this is the first time you are installing C for AIX, follow these steps:

1. Install C for AIX.
2. Enroll licenses with LUM.

If you have a previous version of C for AIX installed, follow these steps:

1. Uninstall the previous version of C for AIX.
2. Install C for AIX.
3. Enroll licenses with LUM.

RELATED REFERENCES

- “System Requirements”
- “Prerequisite Tasks and Conditions” on page 22
- “Installing C for AIX” on page 23
- “Enrolling Licenses with LUM” on page 27
- “Installing Fixes and Upgrades for C for AIX” on page 28
- “Uninstalling C for AIX” on page 29
- “C for AIX Packaging and Filesets” on page 29
- “Upgrading the AIX Operating System” on page 32

System Requirements

Display

SVGA 800x600 (1024x764 recommended)¹

1. Required for the Distributed Debugger, LUM GUI, and online help in HTML and PDF formats.

CD-ROM drive

Required

Mouse or pointing deviceRequired¹**Memory (RAM)**

96 MB minimum (128 MB or higher recommended)

Disk spaceUp to 525 MB¹**Operating System**

IBM AIX Version 4.3.3 or higher

To access the online help

To access the online help, you need to use the Common Desktop Environment (CDE), Adobe Acrobat Reader to view PDF files, and a frames-capable browser such as Netscape Communicator Version 4.04 (or higher).

Prerequisite Tasks and Conditions

Because of the complexity of the C for AIX product, not every prerequisite has been listed in these instructions. Use the preview option to verify and display the required software for your choice of components.

Checking for Required Filesets:

The following items *must* be installed on your system.

Fileset Name	Fileset Description
bos.adt.include	Base Application Development Include Files
bos.adt.lib	Base Application Development Libraries
bos.adt.libm	Base Application Development Math Libraries
bos.net.ncs	Base Network Computing Services
ifor_ls.compat	License Use Management Version 4 Compatibility
ifor_ls.base	License Use Management Version 4 Base

Use the following command to determine if these items have been installed:

```
lspp -h bos.adt.include bos.adt.lib bos.adt.libm \
bos.net.ncs ifor_ls.compat ifor_ls.base
```

Refer to the *AIX Installation Guide* if you need to install these products.

Checking for Other Filesets:

The following optional items are prerequisites for some components.

Fileset Name	Fileset Description
x11.base.rte	AIXwindows Runtime Environment. Install this if you want to invoke the GUI versions of the LUM tools, or if you want to have desktop icons for C for AIX help.
bos.rte.libpthreads	pthreads Library. Required for threaded applications.

Fileset Name	Fileset Description
ipfx.rte	IPF/X Runtime Support. Install this if you want to invoke the GUI versions of the LUM tools.
ifor_ls.base.gui and ifor_ls.client.gui	License Use Runtime Filesets. Install this if you want to invoke the GUI versions of the LUM tools.

Use the following command to determine if these items have been installed:

```
lsipp -h X11.base.rte bos.rte.libpthreads \
      ipfx.rte ifor_ls.base.gui ifor_ls.client.gui
```

Refer to the *AIX Installation Guide* if you need to install these products.

Verifying Space Requirements:

To verify the amount of space needed for the installation, choose the following settings on the SMIT Install Software Products at Latest Level panel before you install C for AIX:

- PREVIEW only? (install operation will NOT occur)
- VERIFY install and check file sizes

The system makes additional resource checks during installation. If you want, you can also specify the following installation option in SMIT:

```
EXTEND file systems if space needed
```

Installing C for AIX

You can install C for AIX in one of two ways:

- Select the individual filesets you want to install on your machine. This allows you to control which components of C for AIX are installed. For more information about the filesets included on the product CD, see “C for AIX Packaging and Filesets” on page 29.
- Install all the filesets found on a CD, as long as they are relevant to your operating system. This option installs all the components found on a CD, giving you the most complete installation of C for AIX possible.

After you have installed C for AIX, you need to enroll your license for the product before using it. For help on installing your license, see “Enrolling Licenses with LUM” on page 27.

Note: If you are upgrading an existing installation of C for AIX, you should uninstall your existing version of C for AIX before installing IBM C for AIX, Version 6.0. See “Uninstalling C for AIX” on page 29 for help on uninstalling the product.

You must have root user access to install C for AIX.

Installing C for AIX by Selecting Filesets

1. Insert the CD into your CD-ROM device
2. At a command prompt, enter `smit install_latest`.
3. Press PF4 or click **List** to display a list of devices.
4. Select the CD-ROM device, then click **OK**.

5. Press PF4 or click **List** to select the filesets that you want to install.
Some of the C for AIX filesets are specific to the AIX version and language environments on your machine. If you select filesets that do not match your version of AIX or your language environments, you will receive a failed install message. We recommend that you read the fileset descriptions closely.

Note: When fileset names differ only by the AIX version that supports them, you should install only the fileset supported by the version of AIX running on your machine. If your machine is running AIX Version 4.3.3, choose *this.aix43.fileset*. If your machine is running AIX 5L™, choose *this.aix50.fileset*.
6. Follow the on-screen instructions to complete this installation.

RELATED REFERENCES

- “After Installing C for AIX” on page 25
- “Installing C for AIX to a Non-Default Directory” on page 25

Installing All the Components of C for AIX

1. Insert the CD into your CD-ROM device.
2. At a command prompt, enter `smit install_latest`.
3. Press PF4 or click **List** to display a list of devices.
4. Select the CD-ROM device, then click **OK**.
5. Click **OK**.
6. Follow the on-screen instructions to complete this installation.

Some of the C for AIX filesets are specific to the AIX version and language environments on your machine. You may receive an error message at the end of the install process.

If you receive an error message at the end of the installation process, check the names and descriptions of the filesets that did not install. File sets not intended for your version of AIX are expected failures. In addition, file sets that require language environments that are not available on your system are also expected failures.

RELATED REFERENCES

- “After Installing C for AIX” on page 25
- “Installing C for AIX to a Non-Default Directory” on page 25

Installing C for AIX Over a Network

If you have a network server installed, you can install C for AIX over a network.

1. At a command prompt, enter `smit install_latest`.
2. In the **Device** field, enter the directory on the client that corresponds to the installation source for on the network server, then click **OK**.
3. (Optional) Press PF4 or click **List** to select the filesets that you want to install.
Some of the C for AIX filesets are specific to the AIX version and language environments on your machine. If you select filesets that do not match your version of AIX or your language environments, you will receive a failed install message. We recommend that you read the fileset descriptions closely.

Note: When fileset names differ only by the AIX version that supports them, you should install only the fileset supported by the version of AIX running on your machine. If your machine is running AIX Version 4.3.3, choose *this.aix43.fileset*. If your machine is running AIX 5L, choose *this.aix50.fileset*.

4. Follow the on-screen instructions to complete this installation.

You can also use the Network Install Manager (NIM) to perform network installs. Refer to the *AIX Network Installation Management Guide and Reference* for more information.

RELATED REFERENCES

- “After Installing C for AIX”
- “Installing C for AIX to a Non-Default Directory”

After Installing C for AIX

After you have installed C for AIX, you need to enroll your license for the product before using it. For help on installing your license, see “Enrolling Licenses with LUM” on page 27.

C for AIX is not automatically installed in `/usr/bin`. To invoke the compiler without having to specify the full path, do one of the following steps:

- Create symbolic links for the specific driver contained in `/usr/vac/bin` to `/usr/bin`.
- Add `/usr/vac/bin` to your path environment variable.

If you use a method other than the AIX `installp` command or the SMIT utility to install C for AIX (such as the non-default install script), the location of the drivers will vary from the default locations above.

Installing C for AIX to a Non-Default Directory

If you use the AIX `installp` command, or the SMIT installation utility, C for AIX is normally installed to `/usr/vac/` directory. You can install C for AIX in a non-default directory using a Perl script provided with the product. This can be helpful if, for example, you want to be able to run two versions of C for AIX.

The non-default install Perl script `vacndi` is provided in fileset `vac.ndi`. Install this fileset first, and then use the script to install C for AIX to an alternate target directory.

You may choose to install just the compiler filesets, or the compiler and the online help. You cannot install IBM Distributed Debugger using the `vacndi` script.

Note: You should only use `vacndi` script to install C for AIX if you are an expert AIX user. To avoid unexpected behavior during installation, do not modify this script.

In order to run the Perl script, you must have the Perl Version 5 runtime environment `perl.rte` installed on your computer. This fileset is shipped with the AIX 5.1 base operating system. To download and install the Perl runtime environment on your AIX 4.3.3 system, refer to the Comprehensive Perl Archive Network Web site at <http://www.cpan.org>.

For detailed information about the `vacndi` script, type `/usr/vac/bin/vacndi -h` on the command line and press Enter.

To install C for AIX to a non-default directory:

1. Navigate to the `/usr/vac/bin/` directory.
2. On the command line, type

```
perl vacndi -d source_path [-e logfile_name] \  
                [-m] [-p target_directory]
```

 - Use *source_path* to specify the directory where the C for AIX filesets are located. This path may also be a mounted CD-ROM drive.
 - Use the `-e logfile_name` option to specify the name and location of the installation log file. Otherwise, the installation log file `vacndi.log` will be stored in your working directory.
 - Use the `-m` option to install just the C compiler files. If you do not specify this option, compiler files and online help files will be installed.
 - Use the `-p target_directory` option to specify the location where the filesets should be copied and expanded. Otherwise, the files will be copied to the `/usr/vac/` directory. If the target directory exists already, you will receive an error message and the installation will stop.
3. Press Enter.

To install a PTF (fix) on the non-default version of C for AIX:

1. Create a text file listing the PTF files you want to install. This text file should contain a name of a single PTF file on each line.
2. Navigate to the `/usr/vac/bin` directory.
3. On the command line, type

```
perl vacndi -d source_path [-e logfile_name] \  
                [-p target_directory] -u ptf_names
```

 - Use *source_path* to specify the directory where the PTF filesets are located.
 - Use the `-e logfile_name` option to specify the name and location of the installation log file. Otherwise, the installation log file `vacndi.log` will be stored in your working directory.
 - Use the `-p target_directory` option to specify the location where the filesets should be copied and expanded. Otherwise, the files will be copied to the `/usr/vac/` directory.
 - Use *ptf_names* to specify the names of PTF packages you want to install.
4. Press Enter.

If you install C for AIX using the `vacndi` script, you will not be able to use AIX tools to uninstall it, or to determine which version of the compiler components is installed. To uninstall the non-default version of C for AIX, delete the directory specified as *target_directory* during the installation. To determine which version of each fileset is installed, review the `.vrmf_history` text file, which is created in the *target_directory* during installation of the compiler and any fixes.

RELATED REFERENCES

- “C for AIX Packaging and Filesets” on page 29

Enrolling Licenses with LUM

Before starting to use C for AIX, you must enroll the appropriate license certificate, using the License Use Management (LUM) software. Three LUM license certificates are provided with each compiler product: a concurrent nodelock license certificate, a concurrent network license certificate, and a simple nodelock license certificate. You should enroll the appropriate certificate for the type of license server you have configured.

Before enrolling a LUM license certificate, ensure that you have installed, configured, and started LUM on your machine.

LUM license file names and locations for C for AIX

Directory	/usr/vac
Concurrent nodelock LUM license file name	cforaix_cn.lic
Concurrent network LUM license file name	cforaix_c.lic
Simple nodelock LUM license file name	cforaix_n.lic

Before you enroll concurrent nodelock and concurrent network LUM license certificates, make sure you have the correct licenses for the authorized users of this program.

Note:

IBM C for AIX, Version 6.0 is licensed based on a charge unit of authorized users for each physical machine running the program.

An authorized user is an individual or specific named user authorized to have access to the program or any portion of the program on a physical machine. The Proof of Entitlement for this program is evidence of your authorization. The number of authorized users (individual or specific named users) allowed to access the IBM C for AIX, Version 6.0 program is determined by the number of authorizations the customer acquires.

Only an authorized user may have access to the program or any portion of the program.

RELATED REFERENCES

- “Enrolling Licenses Using the LUM GUI”
- “Enrolling LUM Licenses From the Command Line” on page 28

Enrolling Licenses Using the LUM GUI

The LUM Basic License Tool runs either from a GUI or a command line interface. You must have root user access to enroll your C for AIX license with LUM.

To enroll a license certificate using the LUM Basic License Tool GUI:

1. To invoke the LUM Basic License Tool, run the **i4blt** command.
 - On AIX 4.3.3, the **i4blt** command is in the /var/ifor/ directory.
 - On AIX 5.1 and higher, the **i4blt** command is in the /usr/opt/ls/os/aix/bin/ directory.

2. Select **Products** > **Enroll Product** from the main menu. The Enroll Product dialog box appears.
3. Click **Import**. The Import dialog box opens.
4. In the **Filter** field, enter `/usr/vac/*.lic` and click **Filter**.
5. Select the correct license certificate file name from the **Files** field, then click **OK**. Information about your C for AIX license should now be displayed in the Enroll Product window.
6. Click **OK**. The Enroll Licenses window opens.
7. (Optional) Fill in the Administrator Information part of the Enroll Licenses window.
8. Fill in the number of valid purchased licenses of the product in the Product Information part of the Enroll Licenses window.
9. Click **OK**. The product should be successfully enrolled.
10. To exit the LUM Basic License Tool, select **Products** > **Exit**.
11. If you have enrolled concurrent network licenses, you must distribute the licenses before starting to use C for AIX. For instructions on how to distribute licenses, see the LUM documentation at <http://www.ibm.com/software/is/lum/library.html>. You can also find simplified instructions for distributing licenses in the README.password file in `/usr/vac`.

Enrolling LUM Licenses From the Command Line

To enroll a license certificate using the LUM Basic License Tool command line interface:

1. Extract the **i4blt** command from the top of the correct product license file.
2. Replace `number_of_lics` in the command with the number of valid purchased licenses of the product.
3. (Optional) Replace `admin_name` in the command with the name of the administrator.
4. Invoke the updated command from the correct directory.
 - On AIX 4.3.3, the **i4blt** command is in the `/var/ifor/` directory.
 - On AIX 5.1 and higher, the **i4blt** command is in the `/usr/opt/ls/os/aix/bin/` directory.

The product should be successfully enrolled.

5. If you have enrolled concurrent network licenses, you must distribute the licenses before starting to use C for AIX. For instructions on how to distribute licenses, see the LUM documentation at <http://www.ibm.com/software/is/lum/library.html>. You can also find simplified instructions for distributing licenses in the README.password file in `/usr/vac`.

Installing Fixes and Upgrades for C for AIX

You can download the latest fixes and upgrades for C for AIX from the **Support** section of the IBM C for AIX web site at <http://www.ibm.com/software/ad/caix/>.

You must have root user access to install fixes for C for AIX.

1. Save the downloaded fix files to a directory accessible by your client machine.
2. Remove the `.toc` file in the download directory after each download to create a new table of contents.

3. (Optional) To view a list of problems resolved by this update, type `smit install_list_problems` on the command line and press Enter.
4. On the command line, type `smit install_latest` and press Enter.
5. Press PF4 to display a list of devices.
6. Specify the download directory as the **INPUT device / directory for software**, then press Enter.
7. To install the full product, press Enter. You can also press PF4 to select the filesets you want to install.
8. Follow the instructions to complete the installation.

Uninstalling C for AIX

You must have root user access to uninstall this product.

1. At the command line, type `smit install_remove` and press Enter. The Remove Installed Software window opens.
2. On the **SOFTWARE Name** line, press PF4. A list of the available software filesets appears.
3. Select all C for AIX filesets, then press Enter.

Some filesets may not uninstall if they are required by other, installed products. See “C for AIX Packaging and Filesets” for details about filesets included with C for AIX.

C for AIX Packaging and Filesets

If you do not want to install all available components, you may choose which filesets to install. In addition, you may specify that any fileset that is a prerequisite to a fileset you selected be installed automatically.

When fileset names differ only by the AIX version that supports them, you should install only the fileset supported by the version of AIX running on your machine. If your machine is running AIX Version 4.3.3, choose *this.aix43.fileset*. If your machine is running AIX 5L, choose *this.aix50.fileset*.

When fileset names differ only by the language code, you should install only the filesets relevant to your desired language and location. The LANG environment variable determines which message catalogs are used. If a catalog for the corresponding LANG cannot be found, English message catalogs are used instead.

Note: In the tables below, *LANG* represents one of the national language codes. For example, US English (ISO) C compiler messages are in `vac.msg.en_US.C` fileset.

RELATED REFERENCES

- “Filesets Required for C Compiler” on page 30
- “Filesets Required for C Compiler Online Help” on page 30
- “Filesets Required for IBM Distributed Debugger” on page 31
- “Filesets Required to Run LUM and SMIT Client GUIs” on page 31
- “C++ Runtime Filesets Required to Run C for AIX” on page 31
- “Optional C for AIX Filesets” on page 32

Filesets Required for C Compiler

The following filesets are included in the C compiler.

Filesets Required for the Command-Line C Compiler

Fileset Name	Fileset Description
vac.C	C for AIX compiler
vac.C.readme.ibm	C for AIX additional information
vac.lic	C for AIX LUM License Files
vac.msg.LANG.C	C for AIX compiler messages
xlopt.lib	XLOPT Optimization Library
xlopt.rte	XLOPT Optimization Runtime
xlopt.tools	XLOPT Optimization Tools
xlsmpt.msg.LANG.rte	XL SMP Runtime Messages
xlsmpt.rte	XL SMP Runtime Library
memdbg.adt	User Heap/Memory Debug Toolkit
memdbg.aix43.adt	User Heap/Memory Debug Toolkit for AIX 4.3
memdbg.aix50.adt	User Heap/Memory Debug Toolkit for AIX 5.1
memdbg.msg.LANG	User Heap/Memory Debug Messages

Filesets Required for C Compiler Online Help

The following filesets are included in C compiler online help.

Filesets Required for C Compiler Online Help

Fileset Name	Fileset Description
vac.Dt.common	C for AIX Desktop Integration Common Files
vac.Dt.help	C for AIX Help Desktop Integration
vac.html.LANG.C	C for AIX Compiler HTML Documentation
vac.html.LANG.search	C for AIX Compiler Documentation Search
vac.html.DBCS.search	C for AIX Compiler Documentation Search Double Byte Common Files
vac.html.SBCS.search	C for AIX Compiler Documentation Search Single Byte Common Files
vac.html.common.search	C for AIX Compiler Documentation Search Common Files
vac.loc.LANG.Dt.help	C for AIX Help Desktop
vac.pdf.en_US.C	C for AIX PDF documentation — English only
IMNSearch.rte	NetQuestion Search Engine
IMNSearch.rte.DBCS	NetQuestion DBCS search engine
IMNSearch.rte.SBCS	NetQuestion SBCS search engine
IMNSearch.rte.httpdlite	NetQuestion web server
vatools.html.help	VisualAge [®] Tools Help
vatools.msg.LANG.html.help	VisualAge Tools Messages Help

Filesets Required for IBM Distributed Debugger

The following filesets are included in IBM Distributed Debugger.

Filesets Required for IBM Distributed Debugger

Fileset Name	Fileset Description
idebug.client.extras	Debugger Interpreted Engine for OS/390®
idebug.client.gui	Debugger Graphical User Interface
idebug.client.olt	Object Level Trace Viewer
idebug.engine.compiled	Debugger Engine for Compiled Languages
idebug.engine.interpreted	Debugger Engine for Interpreted Languages
idebug.help.en_US	Debugger help — English only
idebug.msg.LANG.engine	Debugger Engine Messages
idebug.msg.LANG.olt	Object Level Trace Messages
idebug.rte.hpj	High-Performance Java™ Runtime
idebug.rte.jre	Java Runtime Environment
idebug.rte.olt.Java	Object Level Trace Java Runtime
idebug.rte.olt.client	Object Level Trace Client Controller
idebug.server.olt	Object Level Trace Server

Filesets Required to Run LUM and SMIT Client GUIs

You must have the following IPF/X filesets on your computer before you can run the LUM and SMIT client GUIs. They are provided on the product CD for your convenience.

Filesets Required to Run LUM and SMIT Client GUIs

Fileset Name	Fileset Description
ipfx.adt	IBM Information Presentation Facility Development Kit/6000
ipfx.msg.LANG.adt	IPF/X development kit message and grammar files
ipfx.msg.LANG.rte	IPF/X runtime message and grammar files
ipfx.rte	IBM Information Presentation Facility/6000

C++ Runtime Filesets Required to Run C for AIX

You must have the following C++ runtime filesets on your computer before you can run C for AIX. They are provided on the product CD for your convenience.

C++ Runtime Filesets Required to Run C for AIX

Fileset Name	Fileset Description
xlC.adt.include	C++ Application Development Toolkit
xlC.aix43.rte	C++ Runtime for AIX 4.3
xlC.aix50.rte	C++ Runtime for AIX 5.1
xlC.msg.LANG	C++ Runtime Messages
xlC.rte	C++ Runtime

Optional C for AIX Filesets

The following optional filesets are not required for any C for AIX component.

Optional C for AIX Filesets

Fileset Name	Fileset Description
vac.ndi	C for AIX Non-Default Install Script
vacpp.ndi	VisualAge C++ Non-Default Install Script
vacpp.samples.ansicl	VisualAge C++ Sample Files

Upgrading the AIX Operating System

The minimum operating system requirement for IBM C for AIX, Version 6.0 is AIX 4.3.3. When you upgrade your operating system (for example, to AIX 5.1), you need to consider the impact on C for AIX. There are two possible migration paths.

Recommended migration path:

1. Uninstall C for AIX.
2. Upgrade your operating system.
3. Install C for AIX.

Brief migration path:

1. Upgrade your operating system.
2. Make sure all filesets support the correct level of operating system. For example, if you upgraded from AIX 4.3.3 to AIX 5.1, and you have a fileset named *this.aix43.fileset*, you must install *this.aix50.fileset*.

Note: You must have installed the xlC.rte 5.0.2.1 fileset before you can upgrade your operating system. This fileset contains a packaging change that allows migration installs of AIX 5.1 from an AIX 4.3 system. If you do not install this fileset on your AIX 4.3 operating system, your system will not boot up after you migrate to AIX 5.1. This fix is available on the <http://www.ibm.com/software/ad/vacpp> Web site. Click **Download** and look for "VisualAge C++ for AIX 5.0.2.1 runtime PTF".

Notices

Note to U.S. Government Users Restricted Rights -- use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd. Laboratory
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario L6G 1C7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Programming Interface Information

Programming interface information is intended to help you create application software using this program.

General-use programming interface allow the customer to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification, and tuning information is provided to help you debug your application software.

Note: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and Service Marks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

- AIX
- AIX 5L
- IBM
- Open Class
- OS/390
- PowerPC
- VisualAge

Java and Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

UNIX is a registered trademarks of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Industry Standards

The following standards are supported:

- The C language is consistent with the International Standard for Information Systems-Programming Language C (ISO/IEC 9899-1999 (E)).
- The C++ language is consistent with the International Standard for Information Systems-Programming Language C++ (ISO/IEC 14882:1998).