

HP C/HP-UX Release Notes

Version B.11.23.02

Edition 4

HP-UX Systems



i n v e n t

Manufacturing Part Number: 5990-8185

September 2004

United States

© Copyright 2003 Hewlett-Packard Development Company L.P.

Legal Notices

The information contained herein is subject to change without notice.

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Printed in the United States

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

1 HP C/HP-UX Release Notes

The information in this document applies to the release of HP C compiler, version B.11.23.02, for the HP-UX 11i/11.x operating system.

Announcement

HP C is Hewlett-Packard's version of the C programming language that is implemented on HP Integrity systems. HP C/HP-UX is highly compatible with the C compiler implemented on the HP 9000 Series 300/400 and CCS/C, Corporate Computer Systems C compiler for the HP 3000.

This document discusses the following topics:

- “New Features in Version B.11.23.02” on page 5
- “New Features in Version B.11.11.10” on page 6
- “New Features in Version B.11.11.08” on page 7
- “New Features in Version B.11.11.06” on page 12
- “Documentation Overview” on page 14
- “Installation Information” on page 19
- “Problem Description and Fixes” on page 21

What's in This Version

This section gives an overview of the new features introduced in this version of the HP C compiler.

New Features in Version B.11.23.02

HP C version B.11.23.02 has no changes from version B.11.11.10, except for defect fixes.

New Features in Version B.11.11.10

HP C compiler version B.11.11.10 supports the following new features:

- Function Inlining
- +d Option
- UNIX95 Feature

Function Inlining

Now function inlining happens at all optimization levels, by default. Functions that should be inlined by the compiler need to be tagged with the `__inline` keyword in `-Ae` mode (`inline`, in `-AC99` mode).

+d Option

This option disables function inlining at optimization levels less than `+O3`. Refer to [Function Inlining](#) for more information.

UNIX95 Feature

Previously, if a search directory specified for `-I` or `-L` did not exist or was invalid, the `cc` driver would ignore it silently and the compilation would proceed. This is inappropriate, since an incorrect alternate library or include directory specified on the command line to override a system one will never be detected.

Now, on encountering a search directory on the command line, which, either does not exist or has invalid permissions, an error is flagged and the compilation is aborted. This behavior is applicable only in strict ANSI mode when either `UNIX_STD=95` or `UNIX_STD=98` or `UNIX95=1` has been set in the environment and only when the compiler has been invoked as `c89`.

New Features in Version B.11.11.08

HP C compiler version B.11.11.08 supports the following new features:

- `restrict` Keyword
- `__typeof__` Operator
- `__restrict__` Keyword
- `__volatile__` Keyword
- C99 Mode with `-AC99` Option
- `+We[n1,n2,...,nN]` Option to Flag Warnings as Errors
- Non-Constant Initializers for Aggregates
- `—include <file>` Option
- `+O[no]clone` Option
- `+Olit=[all | const | none]` Option
- `+O[no]memory[=malloc]` Option
- Assigned `goto` Feature
- `__label__` Feature to Locally Declare Labels
- `HPC_DEBUG_COMPAT` Environment Variable

restrict Keyword

This is a C99 feature. The `restrict` keyword tells the optimizer that variables declared as `restrict` cannot have aliases (using pointers). Thus the optimizer can do better alias analysis.

As of the current release, only the keyword is supported without any accompanying optimizations.

__typeof__ Operator

A `typeof`-construct can be used anywhere a typedef name is used. For example, you can use it in a declaration, in a cast, or inside of `sizeof` or `__typeof__`. `__typeof__` is another way to refer to the type of an expression.

This example declares `y` as an array of type of what `x` points to.

```
__typeof__ (*x) y[4];
```

This declares `y` as an array of pointers to characters:

```
__typeof__(__typeof__(char *) [4]) y;
```

__restrict__ Keyword

This keyword can be used to indicate to the optimizer that the associated variable declared with this qualifier will not have aliases (using pointers). This will help in better code optimization.

NOTE __restrict__ qualifier can only be used with pointers.

Examples:

```
int a = 10;  
int *__restrict__ b = &a;
```

__volatile__ Keyword

This is equivalent to the `volatile` keyword.

Usage:

```
__volatile__ int var;
```

C99 Mode with -AC99 Option

This option needs to be used in order to use the C99 specific features. For example, it enables keywords like `restrict`, and `typeof` to be recognized. (Without this option, the keywords `__restrict__` and `__typeof__` need to be used).

+We[n1,n2,...,nN] Option to Flag Warnings as Errors

Using this option causes the compiler warnings `n1`, `n2`, . . . , `nN` to be converted into error messages. If no numbers are specified, then most warnings, except a selected few, are converted into errors.

Non-Constant Initializers for Aggregates

Now it is possible to specify non-constant expressions as initializers in the initialization lists of aggregates, such as arrays and structures. The expressions can be any valid C expressions, such as arithmetic expressions involving variables, and even function calls.

Earlier it was only possible to specify constant expressions as initializers of aggregates.

Example usage:

```
int i, j;  
/* ... */  
int iarr[] = { i, j, i + j, i - j };
```

—include <file> Option

This option causes the compiler to insert <file> at the beginning of a set of translation units, before processing the translation units.

Effectively, it executes implicit #include "<file>" directives for each -include <file> option on the command line, before the very first line of each of the source files specified on the command line. This is a GCC compatibility feature.

Example Usage:

```
$ cc [cc opts] [-include <file>]* [cc opts] <source file(s)>
```

where, [-include <file>]* stands for 0 or more -include <file> options.

All the -include options are processed in the order in which they are written.

+O[no]clone Option

This option allows the user to turn on[off] the cloning facility of the optimizer. Cloning is on by default. It is mainly provided for users who may see a lot of cloning adversely affecting the performance of their code.

If inlining is turned off, cloning is turned off by default. You cannot specify +Onoinline +Oclone.

+Olit=[all | const | none] Option

The +Olit option specifies the type of data items placed in the read-only data section. It can take the values all, const and none.

+Olit=all places all string variables and all const-qualified variables that do not require load-time or run-time initialization in the read-only data section.

+Olit=const places all string literals appearing in a context where `const char *` is legal, and all `const`-qualified variables that do not require load-time or run-time initialization in the read-only data section.

If +Olit=none is specified, no constants are placed in the read-only data section.

+O[no]memory[=malloc] Option

This option enables[disables] memory optimizations. Specifying `malloc` in the list enables[disables] optimizations which consolidate memory allocation procedure calls. This option is disabled by default. It is incompatible with +OopenMP and +Oparallel, and is ignored when these options are in effect.

Assigned goto Feature

Assigned `goto` is a `gcc` compatibility feature, to use labels as values. The address of a label defined in the current function can be got using the unary operator `&&`. The value has type `void *`.

For example:

```
void *ptr;
/* ... */
ptr = && label_foo;
```

To use these values, the user must be able to jump to one. This is done with the computed `goto` statement(1), `goto *EXP;`. For example,

```
goto *ptr;
```

Any expression of type `void *` is allowed.

__label__ Feature to Locally Declare Labels

A local label is simply an identifier. You can jump to it with an ordinary `goto` statement, but only from within the compound statement it belongs to.

The syntax for a local label is given below:

```
__label__ LABEL;
```

or

```
__label__ LABEL1, LABEL2, ...;
```

HPC_DEBUG_COMPAT Environment Variable

HP C does not emit debug information for unused objects (structures, unions, and others) anymore with the `-g` option. For getting the older behavior (emitting debug information for all objects, irrespective of whether they are used in the program or not), the environment variable `HPC_DEBUG_COMPAT` can be set in the environment along with the `-g` option in the compilation command line. It suffices to define this environment variable to have an empty string; it does not need to be set to any particular value.

New Features in Version B.11.11.06

HP C compiler version B.11.11.06 supports the following new features:

- `__FUNCTION__` Predefined Identifier
- `+ub` Option
- UTF-16 Character Support
- `-mt` Option

`__FUNCTION__` Predefined Identifier

`__FUNCTION__` is a predefined pointer to `char` defined by the compiler, which points to the name of the function as it appears in the source program.

NOTE `__FUNCTION__` is same as `__func__` of C99.

The implementation of `__FUNCTION__` adheres to C99 (as per 6.4.2.2 Predefined Identifiers) standards.

`+ub` Option

The `+ub` option treats unqualified `char`, `short`, `int`, `long` and `long long` bitfields as unsigned. This option has no effect on signedness of enum bitfields or on signedness of non-bitfield character.

UTF-16 Character Support

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

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
```

-mt Option

The new `-mt` option enables multi-threading capability without the need to set any other flags, such as `-l` and `-D`. HP C compiler sets the appropriate flags as required.

- `-D_REENTRANT`
- `-D_HPUX_SOURCE`
- `-D__POSIX_C_SOURCE=199506L`
- `libpthread`

Following macros are used to compile multi-thread source code:

- `_REENTRANT` - Required by system header files that provide reentrant functions (suffixed by `_r`).
- `_HPUX_SOURCE` - Required by HP C system header files.
- `_POSIX_C_SOURCE=199506L` - Required by pthread.
- `libpthread.*` - Kernel thread library used on 11.x systems.

Documentation Overview

The HP C/ANSI C compiler and related documentation is available for users of the HP C Developer's Bundle. This documentation is available both online, and in printed copy. Online documentation is located at <http://docs.hp.com>, and is viewable using your any of the available web browser.

The HP C documentation consists of:

- HP C/HP-UX Release Notes
- HP C/HP-UX Programmer's Guide
- HP C/HP-UX Reference Manual
- HP-UX Floating Point Guide
- HP C/HP-UX Online Help
- C Compiler Documentation
- HP C/HP-UX Release Notes

The HP C/HP-UX Release Notes provides release-specific information such as new feature summaries, installation instructions, and known defects. In addition, the Release Notes contains this documentation overview to help you orient yourself regarding available documentation. The release notes are also available online in the text file in the directory:

```
/opt/ansic/newconfig/RelNotes/
```

Printed Documentation

Printed versions of Hewlett-Packard documents are available for ordering. Use the `man manuals` command for details on the documents available for ordering. See also the HP documentation web site <http://docs.hp.com> and the HP C/HP-UX web site at <http://www.hp.com/go/C>.

Listed below are the documents most closely related to use of the ANSI C Compiler:

- *HP C/HP-UX Reference Manual*
Provides reference material for HP C as implemented on HP 9000 systems. This document is based on the ANSI C standard 9899-1990, and it discusses the implementations and extensions unique to HP C on HP-UX. It does not replicate the ANSI C standard and you are referred to the standard, for any finer points not covered.
- *HP C/HP-UX Programmer's Guide*

Contains a detailed discussion about selected C topics. Included are discussions of data type sizes and alignment modes, comparisons between HP C and other languages, and information on 64-bit programming, optimization, threads, and parallel processing.

- *HP-UX Floating-Point Guide*

Describes how floating-point arithmetic is implemented and discusses how floating-point behavior affects the programmer. It also provides reference information about the C and Fortran math libraries.

- HP C Online Help

The C compiler online help is a series of html files containing a combination of reference and how-to information, including the following:

- What is HP C?
- Program organization
- Compiling & running HP C programs
- Optimizing HP C programs
- Parallel options & pragmas
- Data types & declarations
- Expressions & operators
- Statements
- Preprocessing directives
- Calling other languages
- Programming for portability
- Migrating C programs to HP-UX
- Error message descriptions

Before You Begin

Before you begin using HP C Online Help, you should review the following environment variables.

You must set the `DISPLAY` environment variable to a (graphical mode) value that can accommodate the display of an HTML browser. You may set the `BROWSER` environment variable to point to the location of your HTML browser. If you do not set the `BROWSER` environment variable, the compiler automatically runs the browser located in `/opt/ns-navgold/bin/netscape` or `/opt/ns-communicator/netscape`.

You may set the CROOTDIR environment variable to set the root directory of the online help source. If CROOTDIR is not set, the URL of the online help will be `file:/opt/ansic/html/guide/${LOCALE}/c_index.html`; this is assuming that compiler binaries are located in `/opt/ansic/bin`.

Accessing HP C Online Help

To access the online help, on a system where the HP C compiler is installed, enter the following:

```
/opt/ansic/bin/cc +help
```

This command will launch a web browser, displaying the index file for the HP C online help system. The actual file location of the html help is

```
file:/${CROOTDIR}/html/guide/${LOCALE}/c_index.html.
```

If the environment variable CROOTDIR is not set, path will be formed relative to the compiler's root directory; this is usually `/opt/ansic`. See for more information on setting the CROOTDIR environment variable.

If the browser path set by the BROWSER environment variable does not exist, or if the default browser paths `/opt/ns-navgold/bin/netscape` or `/opt/ns-communicator/netscape` do not exist, a message is displayed telling you that the BROWSER environment variable must be set properly.

X-Motif CDE Help is Obsolete

Previous versions of the HP C compiler, when installed on the X-Motif CDE environment, included a CDE version of the online help. This and the accompanying text-based `charhelp` will no longer be updated with the ANSI C compiler. If you want to view online help, please use the HP C HTML Online Help.

Related Documentation

This documentation is available on the HP-UX 11.11 Instant Information CD-ROM and on the web site <http://docs.hp.com>.

- *Parallel Programming Guide for HP-UX Systems*
Describes efficient parallel programming techniques available using HP Fortran 90, HP C, and HP aC++ on HP-UX. This document is also available online at <http://docs.hp.com>.
- *HP-UX 64-bit Porting and Transition Guide*

Describes the changes you need to make to compile, link, and run programs in 64-bit mode. This document is also available online at <http://docs.hp.com> and in the Postscript file `/opt/ansic/newconfig/RelNotes/64bitTrans.bk.ps`.

- *HP PA-RISC Compiler Optimization Technology White Paper*

Describes the benefits of using optimization. This white paper is available online in the PostScript file: `/opt/langtools/newconfig/white_papers/optimize.ps`.

- *HP-UX Linker and Libraries User's Guide*

Replaces the manual *Programming on HP-UX*. To access the *HP Linker and Libraries User's Guide*, use the `ld +help` command, or visit <http://docs.hp.com>.

- *HP Assembler Reference Manual*

Describes the use of the Precision Architecture RISC (PA-RISC) Assembler on HP-UX systems. Describes PA-RISC Assembler directives, pseudo-operations, and how to run the Assembler on HP-UX.

This document is available at <http://docs.hp.com/hpux/dev/index.html#Assembler>.

- *HP-UX Reference Manual*

The reference manual pages, or man pages, are available online (use the command `man man` for more information), and are also available on the CD-ROM. You may also access this manual online by visiting <http://docs.hp.com>.

- HP-UX Software Transition Kit (STK)

Enables the application developer to easily transition software from HP-UX 10.x to either the 32-bit or the 64-bit version of HP-UX 11.0. The kit is available free of charge on the HP-UX 11.0 Application Release CD-ROM, or from the web at <http://www.software.hp.com/STK/index.html>.

HP WDB Debugger Documentation

HP WDB is the HP-supported implementation of the GDB debugger. Refer to the README file in the directory `/opt/langtools/wdb/doc` for information on the documentation provided with the debugger. See also the web site <http://www.hp.com/go/debuggers>.

HP CXperf Documentation

- *CXperf Command Reference*

Provides both introductory and reference information for using the CXPerf performance analyzer.

- *CXperf User's Guide*

Provides information on how to use the CXperf performance analysis tool.

This guide is available at <http://docs.hp.com/hpux/dev/> under the *Performance Tools and Libraries* section.

Installation Information

This section describes the contents of HP C/ANSI C Developer's Bundle for HP-UX. This includes the following topics:

- Beginning Installation
- HP C Developer's Bundle Contents
- Installed Compiler Paths
- Transition Links

Beginning Installation

After loading HP-UX 11.x, you can install your HP C/ANSI C Developer's Bundle. In addition to the C compiler, it contains the HP-UX Developer's Toolkit. To install your software, run the SD-UX `swinstall` command (see `swinstall (1M)`). This invokes a user interface that leads you through the installation.

After installing the HP C/ANSI C Developer's Bundle, install the latest linker patch (PHSS_23440) or its successor. This patch is required by the `+objdebug` option. Without this patch, the option is ignored.

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 operating system package. Most of this information is also available on the web at <http://docs.hp.com>.

HP C Developer's Bundle Contents

The following are the individual components of the HP C Developer's Bundle:

- Auxiliary-Opt: Auxiliary Optimizer for HP Languages (22,465 Kb)
- C-ANSI-C: HP C/ANSI C Compiler (13,604 Kb)
- C-Analysis-Tools: C Language Analysis Tools (339 Kb)
- C-Dev-Tools: C Language Development Tools (1,382 Kb)
- DDE: Distributed Debugging Environment (27,737 Kb)
- DebugPrg: Debugging Support Tools (341 Kb)
- WDB: HP WDB Debugger (3,645 Kb)

- AudioDevKit: HP Audio Developer Kit (479 Kb)
- CDEDevKit: CDE Developer Kit (10,769 Kb)
- ImagingDevKit: HP-UX Developer's Toolkit - Imaging (2,216 Kb)
- X11MotifDevKit: HP-UX Developer's Tool465kit - X11, Motif, and Imake (25,629 Kb)

Be aware that, if you install all the packages, they occupy approximately 126 megabytes of disk space.

Installed Compiler Paths

Most files related to the HP C compiler are installed in the directories `/opt/ansic` and `/opt/langtools`. The installation scripts add the following paths during the installation process:

- `/opt/ansic/bin` and `/opt/langtools/bin` to the login file `/etc/PATH`.
- `/opt/ansic/share/man/%L:/opt/ansic/share/man` and `/opt/langtools/share/man/%L:/opt/langtools/share/man` to the login file `/etc/MANPATH`.

`%L` is replaced by the value of the `LC_MESSAGES` environment variable when the `man` command is executed. It determines the language used for manpage searches. If `LC_MESSAGES` is not set, `%L` defaults to null. See `environ (5)`.

Transition Links

The HP C/ANSI C compiler installation package provides the capability to create and remove transition links from previous HP-UX release locations to HP-UX release 11.x locations. The HP C/ANSI C product installs the ISU transition link table specification files on the system.

The Software Distribution update tool `tlinstall` uses these files to install transition links from previous HP-UX file and directory names to the corresponding HP-UX 11.x file and directory names. To remove these transition links, use the update tool `tlremove`. For more detail, read the update tools manpages. These tools are installed in `/opt/upgrade/bin`.

Problem Description and Fixes

This section details known defect fixes with accompanying CHART defect database numbers (when available) and workarounds for the HP C/ANSI C compiler. This information is provided by release version number, B.11.11.06.

Problems corrected in the final release of the HP C/ANSI C compiler will be referenced in the Software Status Bulletin.

Users with support contracts may access these bulletins and patch information from the HP Support Line database on the World Wide Web located at one of the following URLs:

- <http://us-support.external.hp.com/>
- <http://europe-support.external.hp.com/>

Enhancements and Defect Fixes

This section describes enhancements and defect fixes in the HP ANSI C compiler. Version B.11.11.06 of the HP C/ANSI C compiler has the following enhancements/defects fixed:

- JAGae13383: `-mt` option needed similar to `aCC`
- JAGae20847: UTF-16 support for HP C compiler.
- JAGae13389: Support for `+ub` option to make unqualified bit fields unsigned by default.
- JAGad96950: Support for OpenMP single pragma with clauses.

Workarounds

The following are workaround solutions to previous problems with the HP C/ANSI C compiler:

- `+Onomoveflops` should always be used with the `+FPZ` and `+FPI` floating point options. `+Onomoveflops` prevents floating point instructions from being moved, and replaces integer division by floating point multiply by the inverse.
- If you intend to use GNU style variable argument macros in HP C, note that you can make the concatenation operator `##` prevent syntax errors from occurring when the variable argument comes in as empty (the null string).

However, you can also insert whitespace to the left of the left operand of ## to more accurately specify the intended left operand.

For example, if you use

```
#define foo(f, s...) printf(f, s)
```

Then the macro call

```
foo("Hello world.\n");
```

results in the expansion

```
printf("Hello world.\n",);
```

(Note the comma ",") causing a syntax error.

GNU provides the following workaround for this kind of a situation. If you use:

```
#define foo(f, s...) printf(f, ## s)
```

If the variable parameter *s* is non-null, and if you use:

```
foo("%s %d\n", "Cycles", "1024");
```

The result is:

```
printf("%s %d\n", "Cycles", "1024");
```

as the expansion as you would expect.

However, if *s* is null, this erases the comma to the left of the ## in the macro definition and resulting expansion is:

```
printf("Hello world.\n");
```

Note that the comma is removed.

In order to get the same behavior in HP C, you must insert a space to the left of the comma to make it clear to the preprocessor that the comma is the left operand of the ## operator. Thus your definition for the macro `foo` is:

```
#define foo(f, s...) printf(f , ## s)
```

(Note the space to the left of the ## operator in the macro definition.)

If the space is not inserted, the left operand of the ## operator is understood to be:

```
printf(f ,.
```

Because there is no parameter by that name for `foo`, it is erased.

When specifying declarations within code in the HP C/ANSI C compiler, do

not expect the same behavior in HP aC++. For the example:

```
for(int i = 0; i < j; i ++) int i;
```

Note the lack of a new block opening for the `for` statement. The C++ compiler accepts this form, with warnings, but the C compiler does not. The difference in the way the stack is handled causes the difference in behavior.

Previously, the C compiler did not emit the source file information for the global typedefs. To correct this, use `-y` option along with `-g` when debug info is generated. You can generate debug information by compiling with `+objdebug`.