HP C/HP-UX Release Notes for HP-UX 11.0

HP 9000 Computers

HP C/HP-UX Version A.11.01.00

Manufacturing Part Number: 5967-0041

February 1999

Legal Notice

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Corporate Offices:

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

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

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

Use of this manual and flexible disc(s), compact disc(s), or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

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

PostScript is a trademark of Adobe Systems Incorporated.

UNIX is a registered trademark of The Open Group.

Contents

1. New and Changed Features

New or Changed Options 8
+objdebug Option 8
+W n1[, n2, ... nN] Option 9
+We n1[, n2, ...nN] Option 9
+Ww n1[, n2, ...nN] Option 9
+Olevel=name1[,name2,...] Option 9
+Oreusedir=directory Option. 10
+O[no]promote_indirect_calls Option 11

Improving Shared Library Performance 12
HP_NO_RELOCATION Pragma 12
HP_LONG_RETURN Pragma. 13
HP_DEFINED_EXTERNAL Pragma 14

New Defaults 16

2. Installation Information

3. Documentation Overview

C Compiler Documentation. 20
HP C/HP-UX Release Notes. 20
Printed Documentation 20
HP C Online Help. 21

Related Documentation. 22

4. Problem Descriptions and Fixes

Debugging of Optimized Code. 25

Required Patch. 25

Defect Fixes 26

Contents

Preface

This document summarizes the changes to HP C/HP-UX in version A.11.01.00. This version of HP C is available on HP-UX 11.0.

Please note that product software releases and operating system releases often occur independently of each other. In general, a product software release applies to the immediately preceding or concurrent system release and to all subsequent system releases until the software is revised.

Also, the product software release number (shown above) indicates the version level of the software product at the time that the release notes were issued. Since some product updates do not require documentation changes, it is possible that there may not be a one-to-one correspondence between the version number on your software and the version number of the release notes.

The printed copy of this *Release Notes* is the same as the online file at /opt/ansic/newconfig/RelNotes/ansic.11.01. The printed version is also available at http://docs.hp.com under *Development Tools and Distributed Computing*.

1 New and Changed Features

Version A.11.01.00 of HP C/HP-UX includes support for the following new options:

- +objdebug leaves debugging information in the object files rather than copying it to the executable file.

  NOTE: You must install the latest linker patch (PHSS_16841) or its successor for this option to work. Without this patch, the option is ignored.

- +W suppresses warnings.
- +We changes warnings to errors.
- +Ww enables warnings that would otherwise not be enabled.
- +O*level*=*name* lowers the optimization level for the named functions.
- +Oreusedir=*directory* reduces link time by saving and reusing object files in *directory* rather than recompiling them from intermediate object files.
- +O[no]promote_indirect_calls promotes indirect calls to direct calls to improve performance.

This chapter summarizes these new options and provides additional information on the pragmas HP_NO_RELOCATION, HP_LONG_RETURN, and HP_DEFINED_EXTERNAL which were introduced in an earlier version of HP C/HP-UX.

New or Changed Options

The following command line options are new in this release of HP C/HP-UX.

+objdebug Option

NOTE You must install the latest linker patch (PHSS_16841) or its successor for this option to work. Without this patch, the option is ignored.

When used with `-g`, `+objdebug` leaves debug information in the object files instead of copying it to the executable file at link time, resulting in shorter link times and smaller executables. The default, `+noobjdebug`, copies the debug information to the executable file.

When you specify `-g`, the compiler places symbolic debugging information into the object files. By default, the linker calls pxdb which compacts this debug information & copies it to the executable file. When `+objdebug` was used at compile time, the linker leaves the debug information in the object files. To debug the executable file, the HP WDB debugger must have access to the object files. If you move the object files, use HP WDB's `objectdir` command to tell it where the object files are. (The HP DDE debugger does not support this option.) This option reduces link time and the size of the executable file by avoiding this copying of debug information.

The compile-time default is `+noobjdebug`. If the linker detects any object files that were compiled with `+objdebug`, it will leave the debug information in those files. Any object files not compiled with `+objdebug` will have their debug information copied into the executable file. You can leave debug information in some object files and not in others.

Use the `+noobjdebug` option when linking to explicitly tell the linker to copy all debug information to the executable file, even from files compiled with `+objdebug`.

For information on the HP WDB debugger, see the help file in /opt/langtools/wdb/doc/index.html.

+W *n1*[, *n2*, ... *nN*] Option

This option suppresses the specified warnings, where *n1* through *nN* are valid compiler warning message numbers. See the file /opt/ansic/lib/nls/msg/C/cc.msgs for a partial list of compiler messages. Or use the *dumpmsg*(1) command to see compiler messages in the catalog files in /opt/ansic/lib/nls/msg/C.

+We *n1*[, *n2*, ...*nN*] Option

This option changes the specified warnings to errors, where *n1* through *nN* are valid compiler warning messages. See the file /opt/ansic/lib/nls/msg/C/cc.msgs for a partial list of compiler messages. Or use the *dumpmsg*(1) command to see compiler messages in the catalog files in /opt/ansic/lib/nls/msg/C.

+Ww *n1*[, *n2*, ...*nN*] Option

This option enables the specified warnings. Use it to enable specific warnings when all other warnings have been suppressed with `-w` or `+w3`. Or use it to enable specific warnings that would otherwise be enabled with the +w1 option. It cannot be used to enable `+M`*n* migration warnings. *n1* through *nN* are valid compiler warning messages. See the file /opt/ansic/lib/nls/msg/C/cc.msgs for a partial list of compiler messages. Or use the *dumpmsg*(1) command to see compiler messages in the catalog files in /opt/ansic/lib/nls/msg/C.

+O*level*=*name1*[,*name2*,...] Option

Optimization levels: 1, 2, 3, 4

Default: All functions are optimized at the level specified by the ordinary +O*level* option.

This option lowers optimization to the specified *level* for one or more named functions. *level* can be 0, 1, 2, 3, or 4. The *name* parameters are names of functions in the module being compiled. Use this option when one or more functions do not optimize well or properly. This option must be used with `-O` or an ordinary `+O`*level* option.

This option works the same as the OPT_LEVEL pragma described under “Optimizer Control Pragmas” in the *HP C/HP-UX Programmer's Guide*. This option overrides the OPT_LEVEL pragma for the specified

functions. As with the pragma, you can only lower the level of optimization; you cannot raise it above the level specified in the ordinary +O*level* or -O option. To avoid confusion, it is best to use either this option or the OPT_LEVEL pragma rather than both.

Examples

The following command optimizes all functions at level 3, except for the functions myfunc1 and myfunc2, which it optimizes at level 1.

```
$ cc +O3 +O1=myfunc1,myfunc2 funcs.c main.c
```

The following command optimizes all functions at level 2, except for the functions myfunc1 and myfunc2, which it optimizes at level 0.

```
$ cc -O +O0=myfunc1,myfunc2 funcs.c main.c
```

The following command optimizes myfunc1 at level 2, myfunc2 at level 1, and all other functions at level 3.

```
$ cc +O3 +O1=myfunc1,myfunc2 +O2=myfunc1 funcs.c main.c
```

+Oreusedir=*directory* Option

Optimization levels: 4 or with profile-based optimization

Default: no reuse of object files

This option specifies a directory where the linker can save object files created from intermediate object files when using +O4 or profile-based optimization. It reduces link time by not recompiling intermediate object files when they don't need to be recompiled.

When you compile with +I, +P, or +O4, the compiler generates intermediate code in the object file. Otherwise, the compiler generates regular object code in the object file. When you link, the linker first compiles the intermediate object code to regular object code, then links the object code. With this option you can reduce link time on subsequent links by not recompiling intermediate object files that have already been compiled to regular object code and have not changed.

Note that when you do change a source file or command line options and recompile, a new intermediate object file will be created and compiled to regular object code in the specified directory. The previous object file in the directory will not be removed. You should periodically remove this directory or the old object files since the old object files cannot be reused and will not be automatically removed.

+O[no]promote_indirect_calls Option

Optimization levels: 3, 4 and profile-based optimization

Default: `+Onopromote_indirect_calls`

This option uses profile data from profile-based optimization and other information to determine the most likely target of indirect calls and promotes them to conditionally-executed direct calls. If +Oinline is in effect, the optimizer may also inline the promoted calls. This option is only effective with profile-based optimization, described in "Profile-Based Optimization" in the *HP C/HP-UX Programmer's Guide*.

NOTE

The optimizer tries to determine the *most likely* target of indirect calls. If the profile data is incomplete or ambiguous, the optimizer may not select the best target. If this happens, your code's performance may decrease.

At +O3, this option is only effective if indirect calls from functions within a file are mostly to target functions within the same file. This is because +O3 optimizes only within a file whereas +O4 optimizes across files.

Improving Shared Library Performance

The pragmas described here can improve performance of shared libraries by reducing the overhead of calling shared library routines. All three pragmas should be used together, where applicable, as they depend on one another to a certain extent. You must be very careful using these pragmas because incorrect use can result in incorrect and unpredictable behavior. See the *HP-UX Linker and Libraries User's Guide* for more information on improving shared library performance.

HP_NO_RELOCATION Pragma

This pragma improves performance of shared library calls by omitting floating-point parameter relocation stubs in calls to shared library functions. Put this pragma in header files of functions that take floating point parameters or return floating point data and that will be placed in shared libraries. Putting it in the header file and ensuring all calls reference the header file is one way to ensure that it is specified at the function definition and at all calls.

WARNING **This pragma *must* be at the function definition and at *all* call sites. If the pragma is omitted from the function definition or from any call, the linker will generate parameter relocation code and the application will behave incorrectly since floating point parameters will not be in expected registers.**

Syntax

#pragma HP_NO_RELOCATION *name1*[, *name2*[, ...]]

where *name1*, *name2*, and so forth are names of functions in shared libraries.

Background

Parameter relocation stubs are instructions that move (relocate) floating point parameters and function return values between floating point registers and general registers. They are generated for calls to routines in shared libraries. Relocation stubs are generated when passing floating

point parameters or using a floating point function return in routines in shared libraries. This pragma prevents this unnecessary relocation from being done.

NOTE Do not use this option with functions that use the varargs macros. See the *HP C/HP-UX Reference Manual* or the *varargs*(5) man page for information on the varargs macros.

HP_LONG_RETURN Pragma

This pragma improves performance of shared library calls by using a long return instruction sequence instead of an interspace branch and by omitting export stubs. An export stub is a short code segment generated by the linker for a global definition in a shared library. External calls to shared library functions go through the export stub.

Put this pragma in header files of functions that will go in shared libraries so it is specified at the function definition and at all calls. For functions with floating point parameters or returns, use the HP_NO_RELOCATION pragma along with this pragma.

WARNING **This pragma *must* be at the function definition and at *all* call sites. If the pragma is omitted from the function definition or from any call, the compiler will generate incompatible return code and the application will behave incorrectly.**

Syntax

#pragma HP_LONG_RETURN *name1*[, *name2*[, ...]]

where *name1*, *name2*, and so forth are names of functions in shared libraries.

Background

An export stub is generated by default for each function in a shared library. Each call to the function goes through the export stub. The export stub serves two purposes: to relocate parameters and perform an interspace return.

The HP_LONG_RETURN pragma generates a long return sequence in the export stub instead of an interspace branch. If you also use the HP_NO_RELOCATION pragma (for functions taking floating point parameters), all the code in the export stub is omitted, eliminating the export stub entirely. For functions taking non-floating-point parameters, the HP_LONG_RETURN pragma by itself eliminates the need for export stubs.

NOTE

Using HP_LONG_RETURN without using HP_NO_RELOCATION with floating point parameters, could actually degrade performance by creating export stubs and relocation stubs.

These pragmas improve performance of calls to shared library functions from outside the shared library. Therefore do not use this pragma for hidden functions (see the -h and +e linker options) or for functions called only from within the same shared library linked with the -B symbolic linker option, otherwise this pragma may degrade performance. (See the *HP-UX Linker & Libraries User's Guide* for information on the above mentioned options.)

Do not use this pragma if you compile on PA-RISC 2.0 or later or with the +DA2.0 option since the effect is the default. That is, if no relocations are generated, export stubs are not generated on PA-RISC 2.0 and later, and a long return instruction sequence is generated by default, so this pragma has no effect.

HP_DEFINED_EXTERNAL Pragma

This pragma improves performance of shared library calls by inlining import stubs. Place this pragma at calls to shared library routines along with the HP_NO_RELOCATION pragma (if using floating-point parameters or return values) and the HP_LONG_RETURN pragma.

WARNING

Do not use this pragma at function definitions, only at function calls. Specifying it at function definitions will result in incorrect behavior.

On PA-RISC 1.1, use this pragma only when calling a shared library from an executable file. Using it on calls within an

executable file will cause the program to abort.

Syntax

#pragma HP_DEFINED_EXTERNAL *name1*[, *name2*[, ...]]

where *name1*, *name2*, and so forth are names of functions in shared libraries.

Background

Import stubs are code sequences generated at calls to shared library routines. The import stub queries the PLT (Procedure Linkage Table) to determine the address of the shared library function & calls it. The HP_DEFINED_EXTERNAL pragma inlines this import stub.

NOTE

If your function takes floating-point parameters, you should also use the HP_NO_RELOCATION pragma (if floating point parameters are present). You should also use the HP_LONG_RETURN pragma with this pragma. If you don't, the import stub may be too large to inline.

Use this pragma only on calls to functions in shared libraries. On PA-RISC 2.0, it will degrade performance of calls to any other functions.

New Defaults

Here are the new defaults:

- The `+objdebug` compile-line option is off by default. That is, +noobjdebug is the default.
- The `+W` option is off by default. That is, warnings are enabled.
- The `+We` option is off by default. That is, warnings are not treated as errors.
- The `+Ww` option is off by default. That is, only warnings indicating that code generation might be affected are issued.
- The `+Oreusedir` option is off by default. That is, intermediate object files are not saved between links and are recompiled to object files at every link.
- The `+O`*`level`*`=`*`name* option is off by default. That is, without this option, all functions are compiled at the level specified by the -O option or the ordinary +O *`level`* option.
- The `+O[no]promote_indirect_calls` option is off by default. That is, `+Onopromote_indirect_calls` is the default.

2 Installation Information

Read this entire document, and any other Release Notes or READMEs you may have, before you begin an installation.

The HP C/ANSI C Developer's Bundle for HP-UX contains the following packages:

- 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 (14,217 Kb)
- HPPAK—HP Programmer's Analysis Kit (5,485 Kb)
- BLINKLINK—HP Incremental Linking Facility (1,252 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.

After loading your HP-UX 11.0, 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)). It will invoke a user interface that will lead you through the installation.

NOTE

After installing the HP C/ANSI C Developer's Bundle, install the latest linker patch (PHSS_16841) 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.

NOTE

Most files related to the HP C compiler are installed in the directories `/opt/ansic` and `/opt/langtools`. The installation scripts add `/opt/ansic/bin` and `/opt/langtools/bin` to the login file `/etc/PATH`. They also add `/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).)

NOTE

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`.

3 Documentation Overview

This chapter summarizes the C compiler documentation and various other documents that may be of interest to the C compiler user. Some of this documentation is online, and some may be ordered in printed versions, as described below. Most of this documentation is available on the web site http://docs.hp.com.

The HP C documentation consists of:

- *HP C/HP-UX Release Notes* (this document)
- *HP C/HP-UX Programmer's Guide*
- *HP C/HP-UX Reference Manual*
- *HP-UX Floating Point Guide*
- *HP C/HP-UX Online Help*

The following sections summarize these and other, related documents.

C Compiler Documentation

HP C/HP-UX Release Notes

The *HP C/HP-UX Release Notes*, which you are reading now, provide release-specific information such as new feature summaries, installation instructions, and known defects. In addition, they contain this documentation overview to help you orient yourself regarding available documentation. The release notes are also available online in the text file `/opt/ansic/newconfig/RelNotes/ansic.11.01`.

Printed Documentation

You can order printed versions of a wide variety of Hewlett-Packard documents. Use the `man manuals` command for details on which documents are available for ordering as well as ordering information. 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 some of the documents most closely related to use of the C Compiler.

- *HP C/HP-UX Reference Manual* (92453-90087)

  Provides reference material for HP C as implemented on HP 9000 systems. This document is based on the ANSI C standard 9899-1990, and documents 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 fine points not covered.

- *HP C/HP-UX Programmer's Guide* (92434-90013)

  Contains a detailed discussion about selected C topics for HP 9000 systems. Included are a discussion of data type sizes and alignment modes, some comparisons between HP C and other languages, and information on 64-bit programming, optimization, threads, and parallel processing.

- *HP-UX Floating-Point Guide* (B3906-90006)

  Describes how floating-point arithmetic is implemented on HP 9000 systems and discusses how floating-point behavior affects the programmer. Provides information about the C and Fortran math libraries.

HP C Online Help

The C compiler online help includes examples, tutorial information, error message discussions, and reference information.

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

```
cc +help
```

or

```
/usr/dt/bin/dthelpview -h c
```

and the online help window appears on your display. (Note that this requires that your `$DISPLAY` variable is set correctly. If you do not know how to do this, contact your system administrator.)

You can also click on the ? icon on the HP CDE front panel on a system where HP C is installed and select `HP C Online Reference`.

If you do not have a graphics display or prefer to work with a character-based help interface, enter the command:

```
charhelp
```

and you will get a usage description for this command:

```
charhelp: Usage: charhelp {cc|CC|aCC|f77|ld|dde| -helpVolume file}
```

For help with the HP C compiler, for example, enter:

```
charhelp cc
```

and follow the menus for further direction. The charhelp command is in the directory `/opt/langtools/bin`. For more information, see the man page for *charhelp*(1) (`/opt/langtools/share/man/man1.Z` must be in your `$MANPATH` environment variable).

Related Documentation

This documentation is available on the HP-UX 11.0 Instant Information CD-ROM and on the web site http://docs.hp.com.

- *Parallel Programming Guide for HP-UX Systems* (B6056-90006)

  Describes efficient parallel programming techniques available using HP Fortran 90, HP C, and HP aC++ on HP-UX.

- *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 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 Online User Guide* (B2355-90655)

  Replaces the manual *Programming on HP-UX*. To access the *HP Linker and Libraries Online User Guide*, use the `ld +help` command.

- *HP Assembler Reference Manual* (92432-90012)

  Describes the use of the Precision Architecture RISC (PA-RISC) Assembler on HP 9000 computers. Describes PA-RISC Assembler directives, pseudo-operations, and how to run the Assembler on HP-UX.

- *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.

- 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 the following URL:

  `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.

4 Problem Descriptions and Fixes

For a list of HP C problems and their fixes, see the *Software Status Bulletin* using the reference number 92453-01A.

Users with support contracts may access these bulletins and patch information from the HP SupportLine database on the World Wide Web. The URL is:

`http://us-support.external.hp.com/`

or

`http://europe-support.external.hp.com/`

Debugging of Optimized Code

The current implementation does not support debugging of 64-bit optimized code with the HP DDE or HP WDB debuggers.

Required Patch

After installing the HP C/ANSI C Developer's Bundle, install the latest linker patch (PHSS_16841) or its successor. This patch is required by the +objdebug option. Without this patch, the option is ignored.

Defect Fixes

HP C/ANSI C implements enumeration types and bit fields as signed by default, as specified by the ANSI C standard. As of HP-UX 11.0, HP C/ANSI C in 64-bit mode only incorrectly implemented enumeration types and bit fields as unsigned by default. This defect has been fixed in version A.11.01.00 of HP C/ANSI C. That is, in 64-bit mode, HP C/ANSI C now correctly implements enumeration types and bit fields as signed by default.