HP C/HP-UX Release Notes for HP-UX 11i

HP 9000 Computers

HP C/HP-UX Version B.11.11.02

HEWLETT® PACKARD

5969-7891

March 2001

Legal Notice

Corporate Offices:

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

Contents

Contents

1 New and Changed Features

Version B.11.11.02 includes the following new and changed features:

- OpenMP pragmas and options
- IA-64 compatibility options
- +dumpextern:filename option
- -I- option
- C conditional prefetch support
- Changed INIT and FINI pragmas
- New default for +O[no]promote_indirect_calls
- HP_OPT_DATA on 11x
- Dot Product Optimization

OpenMP pragmas and options

A set of OpenMP pragmas have been added to this version of the HP C compiler for HP-UX. OpenMP is an industry-standard parallel programming model which implements a fork-join model of parallel execution. The HP C OpenMP pragmas included in this release are based upon the OpenMP Standard for C, version 1.0.

To view details about the standard and details about usage, syntax and values, please go to htpp://www.openmp.org/specs. You can download either a postscript (ps) or Adobe Acrobat (PDF) version of the C/C++ Version 1.0 OpenMP standard from this website

+Oopenmp command line option

The OpenMP driver option `+Oopenmp` is added to this release of the HP C compiler. When used with the `+Oparallel` option, `+Oopenmp` prompts the compiler to recognize the OpenMP pragmas and pragmas implemented with this release of the compiler.

NOTE The `+Oopenmp` option is accepted at all optimization levels. However, most of the OpenMP pragmas and pragmas need a minimum optimization level of +O3. To ensure that OpenMP pragmas are recognized, you must specify +O3 on the command line.

When the `+Oopenmp` is used, most of the HP Programming Model (HPPM)pragmas are not accepted.The following HPPM pragmas, howevr, are accepted by the HP C compiler when `+Oopenmp` is issued.

- BLOCK_LOOP
- NO_BLOCK_LOOP
- NO_DISTRIBUTE
- NO_DYNSEL
- NO_PARALLEL
- NO_LOOP_DEPENDENCE
- NO_LOOP_TRANSFORM

- NO_UNROLL_AND_JAM
- OPTIONS
- SCALAR
- UNROLL_AND_JAM

New header file

Every C program that contains OpenMP pragmas and is to be compiled for the current version of HP-UX must include the header file `<omp.h>`. If it does not, the OpenMP pragmas will be ignored. The default path for `<omp.h>` is `/opt/ansic/include`.

C pragmas for OpenMP

A set of OpenMP pragmas for HP C are introduced in this release of the compiler. These include work sharing pragmas and synchronization pragmas.

The following are the OpenMP work-sharing pragmas to be implemented for HP C.

- PARALLEL
- FOR
- SECTIONS
- SECTION
- PARALLEL FOR
- PARALLEL SECTIONS

The following are the OpenMP synchronization pragmas to be implemented for HP C.

- CRITICAL
- BARRIER
- ORDERED

Each of the pragmas to be implemented are discussed in brief below .Only mapping of pragmas is covered here and not the complete functionality of a pragma.

OMP PARALLEL

The OMP_PARALLEL pragma defines a parallel region, which is a region of the program that is executed by multiple threads in parallel.This is the fundamental construct that starts parallel execution.

Syntax

`#pragma OMP PARALLEL` *[clause1, clause2...] new-line structured block*

where *[clause1, clause2...]* indicates that the clauses are optional. There can be zero (0) or more clauses, where clause may be one of the following:

- `PRIVATE (list)`
- `DEFAULT (shared | none)`
- `SHARED (list)`
- `REDUCTION(operator :list)`
- `LASTPRIVATE(list)`

The following clauses are not currently supported in this initial release of the OMP PARALLEL pragma for HP C.

- `IF(scalar-expression)`
- `FIRSTPRIVATE(list)`
- `COPYIN(list)`

OMP FOR

The OMP FOR pragma for HP C identifies a construct that specifies a region in which the iterations of the associated loop should be executed in parallel. The iterations of the loop are distributed across threads that already exist.

This initial release of OMP FOR in the HP C compiler only supports static scheduling. In the case of multiple schedule clauses, the first clause is considered and the rest are silently ignored.

Syntax

`#pragma OMP FOR` *[clause1,clause2...] newline*
for-loop

where *[clause1, clause2...]* indicates that the clauses are optional. There can be zero (0) or more clauses, where clause may be one of the following:

- `LASTPRIVATE (list)`
- `REDUCTION (list)`

- `ORDERED`
- `SCHEDULE (kind [,chunksize])`

NOTE `chunksize` should be an integer constant. Expressions in the place of chunksize are not currently supported.

The following clauses are not currently supported in this initial release of the OMP FOR pragma for HP C.

- NOWAIT
- FIRSTPRIVATE
- PRIVATE

OMP_SECTION/OMP_SECTIONS

The OMP SECTION/SECTIONS pragmas identify a construct that specifies a set of constructs to be divided among threads in a team. Each section is executed once by a thread in the team.

Syntax

```
#pragma OMP SECTIONS [clause1, clause2...]new-line
{
#pragma OMP SECTION new-line
        structured-block
#pragma OMP SECTION new-line
        structured-block
.
.
.
}
```

where *[clause1, clause2...]* indicates that the clauses are optional. There can be zero (0) or more clauses, where clause may be one of the following:

- `LASTPRIVATE`
- `REDUCTION`

The following clauses are not currently supported in this initial release of the OMP SECTION/SECTIONS pragmas for HP C.

- `PRIVATE`

- `FIRSTPRIVATE`
- `NOWAIT`

OMP PARALLEL FOR

The OMP PARALLEL FOR pragma for HP C is a shortcut for an OMP PARALLEL region that contains a single OMP FOR pragma.

Syntax

```
#pragma OMP PARALLEL FOR clause1,clause2... new-line
        for-loop
```

OMP PARALLEL FOR admits all the allowable clauses of the OMP PARALLEL pragma and the OMP FOR pragma. The PRIVATE clause is supported.

OMP PARALLEL SECTIONS

The OMP PARALLEL SECTIONS pragma for HP C is a shortcut for specifying a parallel region containing a single OMP SECTIONS pragma.

Syntax

```
#pragma OMP PARALLEL SECTIONS [clause1, clause2...]new-line
{
 [#pragma OMP SECTION new-line
       structured-block
 [#pragma OMP SECTION new-line
       structured-block
  .
  .
  .
 }
```

OMP PARALLEL SECTIONS admits all the allowable clauses of the OMP PARALLEL pragma and the OMP SECTIONS pragma. The PRIVATE clause is supported.

OMP PARALLEL CRITICAL

The OMP PARALLEL CRITICAL pragma identifies a construct that restricts the execution of the associated structured block to one thread at a time.

Syntax

```
#pragma OMP CRITICAL [(name)] new-line
                          structured-block
```

The critical section `name` parameter is optional. All unnamed critical sections globally map to a single name; this is provided by the HP C compiler.

OMP BARRIER

The OMP_BARRIER pragma synchronizes all the threads in a team. When encountered , each thread waits until all the threads in the team have reached that point.

Syntax

`#pragma OMP BARRIER` *new-line*

The smallest statement to contain a barrier must be a block or a compound statement. OMP BARRIER is valid only inside a parallel region and outside the scope of a OMP FOR, OMP SECTION/SECTIONS, OMP CRITICAL, OMP ORDERED, and OMP MASTER.

OMP ORDERED

The OMP ORDERED pragma indicates that the following structured block should be executed in the same order in which iterations would be executed in a sequential loop.

Syntax

`#pragma OMP ORDERED` *new-line*
structured-block

The OMP ORDERED pragmas must be called within theOMP FOR and/or OMP PARALLEL FOR loops.

OMP MASTER

The OMP MASTER pragma for HP C directs that the structured block following it should be executed by the master thread(thread 0) of the team.

Syntax

`#pragma OMP MASTER` *new-line*
structured block

Other threads in the team do not execute the associated statement.

IA-64 compatibility options

This release of the HP C compiler includes options which provide compatibility support for the IA-64 version of HP C. These include the following:

- -b
- -O/+O
- +O[no]inline:filename

-b

The `-b` option causes the linker to create a shared library rather than a normal executable file. Object files that are processed with this option must contain position-independent code (PIC).

Errors will be issued if `-b` is used with the `-noshared` or `-exec` options.

See the *HP-UX Linker and Libraries Online User’s Guide*, and the *HP C Online Help* for more information about the linker.

-O/+O

The HP C compiler optimization features have been changed so that any compiler optimization option of the form `+O[optionname]` can now be also specified as `-O[optionname]`.

+O[no]inline:*filename*

The `+O[no]inline`:*filename* option enables/disables optimizer inlining for the functions specified by the filename parameter. The filename parameter contains a list of function names, separated by spaces or newlines. The `+O[no]inline` option can occur at optimization levels +O3 and +O4. The default is `+Oinline`.

+dumpextern:*filename* option

The `+dumpextern`:*filename* option performs the same operation as `+Oextern`, but fetches the symbols from a text file, instead of from the command line. This option is different from the `-Bextern`:*filename* option in that the symbols in the textfile are generated automatically by the linker.

The same file is then passed back to the compiler, without the user having to edit the symbols.

C++ name demangling library

The HP aCC name demangling library `libdemangle.a` and the header file `acxx_demangle.h` are included with this release of the HP C compiler. Users who want to take advantage of these aC++ components can now do so without installing the HP aC++ compiler.

For more information on these options go to http://www.devresource.hp.com/STK/man/11.00/acxx_demangle_3.html.

-I- option

The -I- option indicates an optional list of -Idirectory specifications in which a directory name cannot begin with a hyphen (-) character. The -I- option allows you to override the default -Idirectory search-path, called "view-pathing".

Syntax

-Idirs] -I- *[-Idirs]*

[-Idirs] indicates an optional list of -Idirectory specifications in which a directory name cannot begin with a hyphen (-) character.

The -I- option allows you to override the default -Idirectory search-path. This feature is called view-pathing. If -I- is not specified, view-pathing is turned off. This is the default.

Specifying -I- serves a dual purpose, as follows:

- It changes the compiler's search-path for quote enclosed `(" ")` file names in a #include directive in the following manner:
  - — The directory named in the -I option.
  - — The standard include directories /opt/aCC/include and /usr/include. The preprocessor does not search the directory of the including file.
- It separates the search-path list for quoted and angle-bracketed include files.

  Angle-bracket enclosed file names in a #include directive are searched for only in the -I directories specified after -I- on the command-line. Quoted includes are searched for in the directories that both precede and succeed the -I- option.

The standard cc include directories (/usr/include and /opt/ansic/include) are always searched last for both types of include files.

View-pathing can be particularly valuable for medium- to large-sized projects. For example, a project is comprised of two sets of directories. One set contains development versions of some of the headers that the programmer currently modifies. A mirror set contains the official sources.

Without view-pathing, there is no way to completely replace the default -Idirectory search-path with one customized specifically for project development.

C conditional prefetch support

This release of the HP C compiler provides support to users who want to ship a single executable for both PA-RISC 1.1 and 2.0 architectures, including support for the data prefetch feature. With conditional prefetch support, an executable can contain the PA2.0 data prefetch features and still provide accurate answers on PA1.1 machines.

Changed INIT and FINI pragmas

The HP C INIT and FINI pragmas, previously only supported in 64-bit mode, are now available in 32-bit mode as well. The functionality is the same for both modes. Using INIT and FINI pragmas in 32- and 64-bit mode does require the B.11.25 version of the HP-UX Linker. For more information, see the *HP-UX Release Notes for HP-UX 11i*.

New default for +O[no]promote_indirect_calls

The new default for the `+O[no]promote_indirect_calls` is `+Opromote_indirect_calls`. This default is automatically activated at optimization level +O4 if profile-based optimization (PBO) is being performed. For more information about `+O[no]promote_indirect_calls`, see the *HP C Programmer’s Guide* on http://docs.hp.com.

HP_OPT_DATA on 11x

The HP_OPT_DATA pragma, formerly included in the HP C compiler for HP-UX 10.20, has been added to this release of HP C for HP-UX 11i.

Dot Product Optimization

Dot product optimization can be used to achieve more efficient use of the PA-RISC 2.0 architecture, resulting in significant runtime improvements. This optimization is only available at +O3 or +O2 optimization levels. To invoke dot product optimization (also known as sum reduction), you must use the +Onofltacc optimization flag on the command line.

2 Installation Information

This chapter 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)). 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.

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 (14,217 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)

NOTE 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`.

3 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 favorite 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 `/opt/ansic/newconfig/RelNotes/ansic.11.11beta`.

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 Manua*l (B3901-90003)

  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 fine points not covered.

- *HP C/HP-UX Programmer's Guide* (B3901-90002)

  Contains a detailed discussion about selected C topics for HP 9000 systems. 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* (B3906-90006)

  Describes how floating-point arithmetic is implemented on HP 9000 systems 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 will automatically run 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 the previous section “Before you begin” on page 25 for more information on setting the CROOTDIR environment variable.

NOTE 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 will be 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 11.0 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 Online User Guide*

  Replaces the manual *Programming on HP-UX*. To access the *HP Linker and Libraries Online User Guid*e, 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 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. 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.

4 Problem Descriptions 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, including versions B.11.01.20 through B.11.01.22.

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 SupportLine 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/`

Defect fixes

This section describes defect fixes in the HP ANSI C compiler. Error message descriptions for errors referenced in these defect lists can be found in the HP C Online Help, which is bundled with the compiler.

HP C Version B.11.01.22 (March 2001)

The following defects were fixed in version B.11.01.22 of the HP C/ANSI C compiler. Where possible, the associated defect tracking number ("JAGxxxxxxx") is noted with the description of the defect.

- JAGab17723 — The compiler generated an error message that stated that the compiler did not recognize the `#error <varargs.h>` preprocessor directive.
- JAGaa73797 — The C compiler had a limitation of 255 characters for identifiers. This has been increased to 2048 characters.
- JAGac00055 — Signal 11 errors were generated in the compiler due to syntax errors.
- JAGac29093 — Signal 11 errors were generated when variables greater than 1020 bytes were used.
- JAGac39582 — Lint generated errors, complaining about illegal compiler syntax.
- JAGac59791 — Error 1521 (init of global variable fails) was being generated by the compiler and lint.
- JAGad01628 — Signal 11 errors occurred when +DA1.1 flag and +O2 optimization levels used.
- JAGad01887 — Macro expansion was being performed on #pragma STDC arguments. This violated the C9X standard, whcih states that these pragmas should not be subject to macro substitution.
- JAGad03310 — Compiler produced wrong version executable.
- JAGad04122 — Signal 11 errors were generated when +O3 optimization was used.
- JAGad07976 / JAGad0751 — Optimization level 3 (+O3) was failing when used with profile-based optimization (PBO). Symptoms included errors when attempting to link .o files built at +O3.

- JAGad12111 — Warning 2 was being generated when the `+DOosname` option was being used.
- JAGad14074 — Signal 11 errors were being generated by the compiler upon invocation of cpp.ansi.
- JAGad15486 — When compiling a C program in wide mode (+DA2.0W or +DD64), warning 724 was not being emitted in the default warning level.
- JAGad29182 — The HP_OPT_DATA pragma, previously part of the HP C compiler for HP-UX 10.20, was never implemented for 11x until this release of the compiler. See "HP_OPT_DATA on 11x" in this release note for more information.
- JAGad30595 — `+Opromote_indirect_calls` was not being enabled at +O4 using PBO. See "OpenMP pragmas and options" in this release note for more information.
- JAGad34318 — Errors were being generated by the C compiler due to incorrect behavior.
- JAGad37085 — The fusion of intrinsic calls were not on when `+Olibcalls` was used.
- JAGad38191 — Signal 11 errors occurred when optimization level +O4 was used.

HP C Version B.11.01.21 (October 2000)

The following defects were fixed in version B.11.01.21 of the HP C/ANSI C compiler. Where possible, the associated defect tracking number ("JAGxxxxxxx") is noted with the description of the defect.

- JAGad11294 — Signal 11 occurred in ccom.
- JAGad10939 — Compile failed with a Signal 11 when +Olibcalls was specified on the compile line.
- JAGad09629 — Partially initialized cons variables truncated at +O3.
- JAGad07781 — Internal Error:7108 was encountered when compiling with patch PHSS_21223 installed.
- JAGad07501 — Error occurred when compiling with PBO and +O3.

- JAGad04840 — Excessive debugging messages have been removed from the HP C compiler.
- JAGad04768 — The compiler driver was confused by +O0=foo option to turn off optimization in a specific function.
- JAGad04509 — Compiler aborted with a Signal 11 when compiling at +O3.
- JAGad03310 — The compiler produced the wrong version of the executable.
- JAGad03244 — When compiling the SpecInt95 benchmark ijpeg at +O4 and PBO, version B.11.01.20 of the ANSI C compiler died and produced a ucomp Signal 11 error.
- JAGad03160 — The -O default optimization switch produced incorrect code.
- JAGad02692 — Warning 714 occurred when the C compiler did not recognize _asm (built-in function).
- JAGad02497 — Spinlock deadlock panic occurred with an incomplete crashdump state.
- JAGad02322 — Compilation problems occurred with conditional assignments at initialization time.
- JAGad01539 — A large application took five minutes to compile at +O0 but took ~150 minutes to compile with -O.
- JAGad01539 — +O2 optimization took hours longer than +O0.
- JAGad01348 — Changed default update action to "update now" instead of saving to disk.
- JAGad01123 —+O2 optimization was invoked when cc -O +O0 was specified.
- JAGad00829 — Signal 11 (Error 7815) occurred with long path names and +objdebug.
- JAGad00760 / JAGad00073— Error 5219 occured with _asm LDBS when compiling with +DA2.0W on long register result.
- JAGad00080 — Optimizer warnings were displayed when compiling in 64-bit mode.
- JAGac59702 — IOC_IN macro in sys/ioctl.h caused Warning 541 (int overflow).

- JAGac40882 — Bad code for +Oentrysched was found.
- JAGac29093 — Signal 11 occurred when variables of more than 1203 bytes were used.
- JAGab25923 — Error 8901 occurred with +O3 +Oparallel using a loop stride of 2.
- JAGab16729 — Signal 11 occurred when compiling with -O -g.
- JAGaa68045 — DA2.0W +O4 created duplicates of inline destructors.
- There was an error in the HP C HTML Online Help in version B.11.01.20 of the HP C compiler. The pragma syntax described in the section “Compiling & Running HP C Programs” contained a string of superfluous information that *preceded* the actual pragma syntax. This information was displayed as follows:

  ```
  font=* charset=’iso8859-1’ code=35
                   TeX=’\num ’ descr=’[num]’
  ```

  Note that syntax for each pragma should begin with the string `pragma` followed by the pragma name, such as `pragma LIST { ON }.`
- xdb, the HP-UX debugging tool, has been replaced by HP’s gdb product WDB. All references to xdb in the HP C Online Help and HP C documentation should actually refer to WDB. To obtain more information about WDB, visit the HP WDB web site at http://www.hp.com/go/wdb.

HP C Version B.11.01.20 (June 2000)

The following defects were fixed in version B.11.01.20 of the HP C/ANSI C compiler. Where possible, the associated defect tracking number ("JAGxxxxxxx") is noted with the description of the defect.

- JAGaa78850 — The binary created by compiler does not work the same as previous compiler binaries.
- JAGaa78856 — If an inline function contained a simple boolean expression where one operand was a bool, bad code generation sometimes occurred.
- JAGaa79841 — Execution of a big switch statement went into a loop when compiled with +DD64 at +O1 or higher.
- JAGac39817 — Duplicate of JAGab16729. Signal 11 occurred when compiling with -O -g.

- JAGaa78566 — Boolean A = B | | A, or A = B && A failed.
- JAGaa78821 — Incorrect debug information was generated.
- JAAGaa78848 — Incorrect code generated for accessing data in text subspace.
- JAGab78867 — pragma HP_ALIGN NOPADDING did not work in 64-bit mode.
- JAGaa73985 — +pa failed with Comdat and .tools_routines.
- JAGaa7855 — C application was hanged at +O2.
- JAGaa78839 — Incorrect error message about pc relative access for 64-bit linker.
- JAGaa78840 — Incorrect code generated for pc rel fixup.
- JAGaa78665 — The compiler generated poor code when comparing values against "sizeof" operators in 64-bit mode on HP-UX 11.0.
- JAGab15575 — The compiler was not writing all the necessary information into the elf .o, resulting in less information in the elf .o than in the som .o file.
- JAGab81693 — Unexpected signal 11 when compiling Spec CPU2000 tools.
- JAGaa95493 — N-class not recognized as being PA-RISC 2.0.
- JAGab70505 — When compiled with -O +DA1.0 +DS1.1, ccom terminated with a Signal 11.
- JAGab68749 — PBO data was ignored even when the source/timestamp was not changed.
- JAGaa78598 — When compiling the frontend with +DA2.0 and PBO, an alignment trap in make_space occurred.
- JAGab75118 — When compiling a Sybase file with PBO and +ESlit, the compiler emitted Error 7108.
- JAGaa93548 — Invalid code was generated at +O2 for fields in union.
- JAGab68267— When compiling some code in Sybase, the C compiler emitted bad code.
- JAGab82656 — A Signal 11 occurred when when compiling with +pal and +O2.
- JAGab14297 — Nested Triadic expressions cause compiler to core

dump.

- JAGaa68621 — Application optimized with +O3/+O4 works on HP-UX 10.20 but failed under 11.0.
- JAGab43749 — Patch PHSS_18301 has install problems.
- JAGab31896 — Compiling at +O2 aborted with Error 1405 Signal 11 but worked successfully with +O1.
- JAGab71490 — When applications were compiled at +O2, the compiler emitted Error 1405.
- JAGab72754 — Bad TLS pointers were generated when compiling with +DA2.0W +O2 +Optrs_strongly_typed.
- JAGab70528 — SPEC CPU2000 produced wrong answers at +O3 but ran fine at +O2 and +O4.
- JAGaa78740 — nm command did not print proper sizes for elf binaries.
- JAGaa71343 — Compiler Internal Error 5172 occurred: "Backend Assert ** Block number not in lex or static mem tables."
- JAGaa68298 — Signal 6 returned by ccom when compiling with +O3.
- JAGaa72645 —When the compiler autoparallelized a loop, it was not properly assigning an element in a 200 element array.
- JAGaa68020 — A ucomp error at line 0 occurred: "cannot mmap file -errno."
- JAGaa71413 — An error stride of unbounded shape occurred at +O3+Oparallel.
- JAGaa68356 — Signal 11 occurred while linking with -P (PBO).
- JAGaa68329 — Aborts occurred with Joining expressions that had different sizes.
- JAGaa72653 — There was a problem with the loop unrolling and real numbers when run at +O3.
- JAGaa71957 — The warning that variables need to be in LOOP_PRIVATE were not always generated.
- JAGaa68349 — Public Domain application failed at runtime with some noticeable output differences.
- JAGaa68307 — The compiler aborted when compiling with cc +o.

- JAGaa93376 — 64-bit optimizer generated poor code, invalid code for field copy.
- JAGab71106 — Signal 11 occurred when using +o.
- JAGab69983 — ccom entered an infinite loop when using the combination of +Onolimit +Odataprefetch +O3 +Oprocelim +Olibcalls.
- JAGab68751 — When compiling at +O3, a Signal 11 occurred.
- JAGab68756 — When compiling at +O3 on a Sybase file, a Signal 11 occurred.
- JAGab68748 — When compiling at +O3 on a Sybase file, a ucomp error occurred.
- JAGaa78819 — Error 7831" "Bad symbol type" was emitted when the +DA2.0W and +ESlit options were used together.
- JAGaa79850 — Compiling PRO/E with the ANSI C compiler at optimization levels +O3 and +O4 resulted in Error 8901.
- JAGaa92788 — PRO/E header files failed to compile.
- JAGaa94874 — Constant expression overflow messages are mistakenly supporessed during the build of the OS kernel.
- JAGab15384 — Warning 555 was unnecessarily emitted when the application source contained the same struct name as the API name.
- JAGab16468 — +O[no]extern=putchar option is not recognized when the +Olibcalls option is used. This resulted in applications crashing with illegal instructions.
- JAGab16732 — Bad debug information was generated for function arguments, which are pointers to integral (char, int) types.
- JAGab17869, JAGab17870, JGab17871 — ANSI C filesets did not specify attributes for architecture, machine_type, os_name, os_release, and v.
- JAGab71889 — Error 1404: "can't find the process" and archive linking errors occurred when attempting to build an ISV application.
- JAGab68316 — swverify directory errors occurred.
- JAGab69121 — Stdarg failed on struct-by-value and ref when used with +DA2.0W.
- JAGab73312 — The ANSI C compiler aborted with segmentation

violations when using the combination of the +o +L +DA2.0W options.

- JAGab84846 — lint(1) message files could not be opened because LANG environment variable was not set to C.
- JAGac29525 — Default +ESlit behavior emitted an unexpected Warning 611 message.
- JAGac56982—In the 64-bit environment, using the +ESlit option with the +DA2.OW option resulted in an errors. The combination of these options tells the code generator to locate the const data in the readonly code segment so that it becomes an error in the 64-bit environment. In the 32-bit environment, however, this error occurs silently. The linker will only detect and display error information for this inconsistency on 64-bit addressing.
- JAGac59791 — Global variable initialization failed, resulting in Error 1521.
- JAGac95076 — The compiler prohibited the combination of +P and -g, resulting in Warning 446: "+P and -g are mutually exclusive. +P option ignored."
- `+Onoinline` has been corrected so that, when specified, inlining does not occur.
- Inlining previously dropped the register storage class of a variable. This has been corrected.
- HP 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.

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 and replaces integer division by floating point multiply by the inverse. If you are goint to enable
- 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, if for example, 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 gone.

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 `+Oobjdebug`.

- The `+Olibcalls` transformation in the HP C compiler has been changed so that the following information in the *HP C/HP-UX Programmer's Guide* is no longer valid:

  “Calls to `setjmp()` and `longjmp()` may be replaced by their equivalents `_setjmp()` and `_longjmp()`, which do not manipulate the process's signal mask.”

  Note that all other tranformations of `+Olibcalls` are unaffected by this change.