

**HP Fortran v2.8
for HP-UX 11.0/11i v1 and 11i v2
Release Note**



**Manufacturing Part Number: 5990-6773
0304**

U.S.A.

© Copyright 1979-2003 Hewlett-Packard Development Company, L.P.

Legal Notices

The information in this document is subject to change without 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 held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. 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.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304
U.S.A.

Use of this release note 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.

Copyright Notices. © Copyright 1979-2003 Hewlett-Packard Development Company, L.P., all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

© Copyright 1979, 1980, 1983, 1985-93 Regents of the University of California

Trademark Notices. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

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

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the U.S. and other countries.

Parts of this compiler were generated by the iburg code-generator generator, described at <http://www.cs.princeton.edu/software/iburg>.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries of the United States and other countries.

1 HP Fortran v2.8 for HP-UX 11.0/11i v1 and 11i v2

Announcement

HP Fortran is a modern, powerful mathematical and scientific language that supports array-handling, data-abstraction, and data hiding.

This Release Note describes HP Fortran v2.8 for HP-UX 11.0/11i v1 and 11i v2. HP Fortran v2.8 is supported on Hewlett-Packard servers and workstations running HP-UX 11i Version 2.0 operating systems for PA-RISC and Itanium®-based systems. HP Fortran v2.8 supports both 32- and 64-bit addressing.

What's in This Version

HP Fortran v2.8 contains object-oriented feature optimizations, including:

- Full Fortran 95 compiler (based on International ANSI/ISO standards)
- *Changes / additions to directives and options:*
 - +Ofaster (revised)
 - ipo
 - optimize openmp [on|off]
 - +Oautopar (revised description)
 - +Ofast (changed implied options)
 - +DOosname
- *NEW:* Resolving issues with unaligned data
- Full OpenMP v2.0
- Linking HP Fortran 90 routines
- Support for the HP WDB Debugger
- Support for math intrinsic inlining
- Standard Fortran library

Recent changes/additions to directives and options

The following reflects recent changes/additions to directives and options:

+Ofaster

For Itanium®-based systems only: Selects the +Ofast option at optimization level +O4. Equivalent to +Ofast +O3 -ipo.

For PA-RISC systems only: +Ofaster drops to +O3.

-ipo (For Itanium®-based systems only)

Enables interprocedural optimizations across files. Object files produced using this option contain intermediate code in IELF format. At link time, ld automatically invokes the interprocedural optimizer u2comp if any of the input object files is an IELF file.

This option is ignored at optimization levels +O1 and +O2. It is enabled by default when +O4 or +Ofaster are used.

optimize openmp [on | off]

- \$optimize openmp on
\$optimize openmp off
- Must be placed between subroutines
- Turns on or off recognition of OpenMP directives

The behavior of this directive is similar to fsplit'ing a file and selectively choosing +Openmp or +Onoopenmp per each file.

+Oautopar

- +Oparallel inhibits +O4.
- If automatic parallelization is desired along with +O4, then use +Oautopar (and omit +Oparallel).
- Is now linked in libcps.

+Ofast

- +Ofast no longer implies +Oshortdata.
- +Ofast now passes the following options to the linker:
 - Wl, +pi, 4M
 - Wl, +pd, 4M
 - Wl, +mergeseg (on Itanium®-based systems only)

+DOosname

- Sets the target operating system for the compiler.
- Intended for enabling optimizations that are not backward compatible.
- Can be used at any level of optimization.
- PA-RISC—based system recognized values for osname are 11.0 and 11.0EP9806. Itanium®—based system recognized values for osname are 11.20, 11.22, and 11.23. +DOosname selects optimizations that are compatible with operating system version osname and later.
- Default value for osname is the operating system version of the computer on which the compiler is invoked.

What's in This Version

+Oprofile=collect[:<qualifiers>]

In the absence of qualifiers, +Oprofile=collect is the same as +I. <qualifiers> are a comma-separated list of profile collection qualifiers. Supported profile collection qualifiers:

- `arc`—collect arc counts (synonym for +Oprofile=collect). This is the default value.
- `stride`—collect stride data.
- `all`—collect all types of profile data. This is a synonym for +Oprofile=collect:arc, stride.

Interactions and dependencies are not changed with respect to +Oprofile=collect.

+Ofailsafe

This option is now supported for the Itanium® processor family.

HP\$OPTIMIZE

HP\$OPTIMIZE, as listed below, is now supported for Fortran 90.

- `!HP OPTIMIZE ON`
- `!HP OPTIMIZE OFF`
- `!HP OPTIMIZE level 1`
- `!HP OPTIMIZE level 0`
- `!HP OPTIMIZE level 2`
- `!HP OPTIMIZE level 1 on`
- `!HP OPTIMIZE level 0 on`
- `!HP OPTIMIZE level 2 on`
- `!HP OPTIMIZE level 1 off`
- `!HP OPTIMIZE level 0 off`
- `!HP OPTIMIZE level 2 off`
- `!HP optimize assume_no_parameters_overlaps on`
- `!HP optimize assume_no_parameters_overlaps off`

Binary I/O Conversions (*for Itanium®-based systems*)

This feature enables the specification of a binary conversion to happen to all unformatted sequential data on reads and writes to various units. Both the OPEN and INQUIRE commands accept the CONVERT argument.

Allowable conversions

There are two types of allowable conversions:

- Native—the default, and no conversions will occur. The user may also specify “Big_Endian”.
- Little Endian—will convert data from Little_Endian format to the Native format while reading it in and back to Little_Endian format when writing it out to disk.

When the CONVERT argument is specified to the OPEN statement, all I/O to that unit will undergo the specified conversions.

When the CONVERT argument is specified to the INQUIRE statement, a string is returned that describes the conversions being done at runtime for a given unit.

In addition to specifying the CONVERT parameter to the OPEN statement, there are two other methods for specifying the type of conversion to take place:

- If the environment variable FORT_CONVERT<unit> (where <unit> is the unit number to convert) is specified to either of the accepted conversion types (Native or Little Endian), that unit will undergo conversion as if the CONVERT parameter was passed to each open statement referring to that unit. Many different environment variables may be required to convert every different unit in a program.
- There is a global FORT_CONVERT environment variable without a unit number, which may be used to refer to all of the units in a program.

Specifying a binary conversion

There are three ways to specify what file/unit should be converted, and which conversion to apply (in priority, with highest priority listed first):

- Environment variable: “FORT_CONVERT<unit_num>”—specifies unit number
- Environment variable: “FORT_CONVERT”—specifies for all I/O
- Argument to OPEN: “CONVERT=”—specifies for this particular file opened

Typically, code is written via the CONVERT=argument to OPEN and INQUIRE. Therefore, this option was given the higher priority status to allow overriding “normal” coding via the environment variable.

NOTE The CONVERT parameter to OPEN is only available via f90 version 2.7 or later, and is only accepted for UNFORMATTED files.

Resolving issues with unaligned data

The following suggestions will aid with issues regarding unaligned data:

For Itanium®-based applications

As a quick fix, add the following call to the main program:

```
call allow_unaligned_data_access( )
```

Then, link in an additional library that fixes these alignment issues at runtime:

```
-lunalign
```

Longer term, try compiling with the following flag:

```
+allow_unaligned
```

An example would be:

```
PROGRAM bs30
INTEGER*2 i2(10)
INTEGER*4 i4
EQUIVALENCE (i2(2), i4)
DATA I4 /z'FFFF0000'/
i4=12
    call allow_unaligned_data_access()

print *,i4
END
```

Then compile with:

```
/opt/fortran90/bin/f90 foo.f -lunalign +noppu
```

If you need to drop the +noppu, then add an alias directive as follows:

```
!$HP$ALIAS allow_unaligned_data_access='allow_unaligned_data_access'
```

```
PROGRAM bs30
INTEGER*2 i2(10)
INTEGER*4 i4
EQUIVALENCE (i2(2), i4)
DATA I4 /z'FFFF0000'/
i4=12
    call allow_unaligned_data_access()
```

```
print *,i4  
END
```

Then compile with:

```
/opt/fortran90/bin/f90 foo.f -lunalign
```

NOTE Nothing new is required to use the aforementioned `-lunalign` runtime fix approach; however, you do need the latest patch for the `+allow_unaligned` compile flag.

If the misaligned data might be passed to routines which are not compiled with `+allow_unaligned` (for example, Fortran runtime IO and F90 libraries), then having the call will allow the runtime fixes to happen which will prevent bus errors in your application.

To realize optimum performance, try and measure the following two options:

1. Compile with `+allow_unaligned` and insert the call to `allow_unaligned_data_access()`
2. Only insert the call to `allow_unaligned_data_access()`

Option #1 may perform faster with very few references to misaligned data and a handler is invoked for these misaligned references. #2 may perform faster with many references.

Full OpenMP v2.0

HP Fortran now supports full OpenMP v2.0. OpenMP is a portable, scalable model that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from the desktop to the supercomputer.

More information

Complete information about this feature is available in the *Parallel Programming Guide for HP-UX Systems*.

Linking HP Fortran 90 routines

When calling HP Fortran 90 routines on the HP 9000 Series 700/800, you must include the appropriate run-time libraries by adding certain arguments to the `aCC` command when linking your program. These arguments depend on how the Fortran 90 routines were compiled.

32-bit PA-RISC 1.1 code

```
-L/opt/fortran90/lib/ -lF90 -lisamstub
```

[HP Fortran v2.8 for HP-UX 11.0/11i v1 and 11i v2](#)

[What's in This Version](#)

32-bit PA-RISC 2.0 code

`-L/opt/fortran90/lib/pa2.0/ -lF90 -lisamstub`

64-bit PA-RISC 2.0 code

`-L/opt/fortran90/lib/pa20_64/ -lF90 -lisamstub`

HP WDB Debugger

The HP WDB debugger is an HP-supported implementation of the GDB debugger. It supports source-level debugging of object files written in HP Fortran, HP C, and HP aC++ on HP-UX Release 10.20 and later. For additional information, please visit <http://www.hp.com/go/wdb>.

HP Caliper v3.0 (for Itanium®—based applications only)

HP Caliper 3.0 is a new general-purpose performance analysis and performance improvement tool for Itanium®-based applications bundled with HP Fortran, HP ANSI C, and HP aC++. HP Caliper helps you analyze and improve the performance of your native Itanium®-based programs in three ways:

- A simple and quick way to optimize the performance of your program by providing information for compiler profile-based optimization.
- Commands to measure the overall performance of your program.
- Commands to drill down to identify performance parameters of specific functions in your program.

A significant advantage of HP Caliper is that it requires no special compilation, link options, or libraries for your Itanium-based programs.

HP Caliper dynamically measures performance on:

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

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

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

Documentation

HP Caliper 3.0 includes the following documentation:

- On-line User Guide in HTML format available in the `/opt/caliper/doc` subdirectory
- Printable User Guide in PDF format
- Command-line option summary available with `--help` option
- The *caliper (1)* man page

Known Problems and Workarounds

This section details known problems and workarounds for HP Fortran.

- When very large arrays are declared to be local to a subroutine, the arrays are privatized. This means that the arrays are allocated in stack space. **Workaround:** The default stack size for an OPENMP thread is 8 megabytes. This can be changed by setting the environment variable `CPS_STACK_SIZE`. For example, `setenv CPS_STACK_SIZE 20000000` changes the stack size per thread from 8MB to 20MB.
- HP Fortran only accepts the `!$ALIAS` directive, not `$ALIAS`.
- A program compiled with `“+DA2.0W +check=all +fp_exception”` results in a divide by zero trap during a subroutine call.
- Compiling an array slice assignment at `+O0` results in an internal compiler error.
- Dummy arguments in multiple entry routines are not correctly handled by the WDB debugger.
- Printing a dummy argument that is a CHARACTER in 64-bit wide mode (`+DA2.0W`) does not work in the WDB debugger.
- When compiling, the `+ppu` switch is enabled by default. To access C routines, use the ALIAS directive to map the C routine name to itself. For example:

```
!$HP ALIAS getrusage=`getrusage`
```

If the ALIAS directive is not used, ``getrusage`` is mapped to ``getrusage_`` in the object file.
- With `+DD64`, memory addresses are 64-bit values. This allows common blocks and dynamically allocated memory to exceed 32-bit address limits. This feature is restricted by the available virtual memory on the system where the application is run.

Common blocks greater than 2GBytes are automatically placed into a huge data segment and initialization is performed at program start-up.

Workaround: To force smaller common blocks into the huge data segment, use the `+hugecommon` and `+hugesize` compiler switches.

NOTE	When using initialized huge common blocks, large repeat counts are represented compactly to help reduce object file size.
-------------	---

- When creating 64-bit shared executables (such as when +DD64 is specified) the +Z option is on by default. The +Z option (equivalent to +pic=long) is the only PIC option supported for 64-bit executables. If the +z option (equivalent to +pic=short) is specified when creating 64-bit code, it instead maps to +Z.

Workaround: To not generate position independent code for 64-bit executables, specify the following option:

-Wl,-noshared

- Mixing the standard Fortran I/O operations with BUFFER IN and BUFFER OUT I/O on the same logical unit number can confuse the input stream (READ) or corrupt the data file (WRITE).
- SoftBench support is not available.
- Using the ON statement at optimization levels 2 and above is restricted. When compiling at optimization level 2 or above, the optimizer makes assumptions about the program that do not take into account the behavior of procedures called by the ON statement. Such procedures must therefore be “well-behaved”; they must meet the following criteria:
 - The ON procedure must not assume that any variable in the interrupted procedure or in its caller has its current value. (The optimizer may have placed the variable in a register to be stored there until after the call to the interrupted procedure is complete.)
 - The ON procedure must not change the value of any variable in the interrupted procedure or in its caller if the effect of the ON procedure is to return program control to the point of interrupt.

NOTE These restrictions do not apply if you compile at optimization levels +O0 and +O1.

- The +DS compile-line options do not accept any PA-RISC architecture models as arguments because the HP Fortran compiler for Itanium®-based architectures does not support the PA-RISC architecture.
- Pointer arithmetic on Cray-style pointers has different semantics on Hewlett-Packard machines than on some Cray machines. HP Fortran implements Cray pointers as BYTE pointers, while some Cray machines use word addressing.
- Large source files occasionally cause internal errors in the HP Fortran compiler.
Workaround: If this occurs, split the file using fsplit and recompile.
- The previous versions of Fortran occasionally contained a 'wrong answer' problem that is now caught by a compiler assert function at +O3 +Oparallel.

Known Problems and Workarounds

Workaround: To avoid this, use +O2 optimization instead of +O3.

Restrictions in HP Fortran

This section lists restrictions you should observe when using the HP Fortran compiler and when using HP Fortran language features in your programs.

+FPI

+FPI will not work when calling any type of runtime libraries or intrinsics.

Object files generated at optimization level 4

Object files generated by the compiler at optimization level 4, called intermediate object files, are intended to be temporary files. These object files contain an intermediate representation of the user code in a format that is designed for advanced optimizations. Therefore, Hewlett-Packard may be required to change the format of these intermediate files in any compiler release or patch. Use of intermediate object files should be limited to the compiler which created them. For the same reason, intermediate object files should not be included into archived libraries that may be used by different versions of the compiler. The compiler will issue an error message and terminate when an incompatible intermediate object file is detected.

+Oparallel_intrinsics

The +Oparallel_intrinsics option links in the parallel version of many of the Fortran intrinsics located in libF90_parallel. However, libF90_parallel for Itanium®-based systems is still only a serial library. It will be parallelized in a future patch.

“OUT OF FREE SPACE” Error

The *HP Fortran Programmer's Reference* states that the IOSTAT= and ERR= specifiers return error 913 (OUT OF FREE SPACE) when the I/O library attempts to use more memory than is available. However, these specifiers do not detect all instances of error 913, especially those caused by memory allocation failures in the I/O library.

NODEPCHK ignored

The NODEPCHK directive is sometimes ignored. The symptom was that programs which use the directive would not parallelize as well as expected.

Restrictions in HP Fortran

To parallelize your code, use another directive such as `loop_parallel`. In loops defined with `NODEPCHK`, we recommend that you not invoke parallelism.

System Compatibility Information and Installation Requirements

This section describes system requirements for the installation and running of HP Fortran v2.8.

Software requirements

The HP Fortran v2.8 compiler requires approximately 264 megabytes (MB) of disk space. This includes approximately 172 MB for the compiler, with remaining space for other components such as the debuggers.

Hardware requirements

HP Fortran v2.8 is supported on hardware running HP-UX 11.0 and 11i.

Operating system requirements

HP Fortran v2.8 recommends that HP-UX 11i Version 2.0 be installed.

Installation requirements

To install HP Fortran v2.8 for HP-UX 11i Version 2.0, follow these steps:

1. Start the swagentd daemon if it is not already running. Enter the following command in the console window:

```
/usr/sbin/swagentd
```

2. Use the swinstall command to install the desired product(s) from the CD-ROM.
3. Enter the following command to install the Fortran compiler on HP-UX 11i Version 2.0:

```
/usr/sbin/swinstall -x rpc_timeout=9 -x mount_all_filesystems=false -s  
/dev/dsk/c0t0d0 [B3907DB | B3909DB]
```

Support information on HP Fortran

HP customers who have purchased support contracts can find a list of HP Fortran language problems and their fixes in the current *Software Status Bulletin* (SSB). This information can be found by referencing the following product numbers:

System Compatibility Information and Installation Requirements

- B3907DB—HP Fortran Workstations
- B3909DB—HP Fortran Servers

To display the product number and the release version of your HP Fortran compiler, execute this HP-UX command:

```
what /opt/fortran90/bin/f90
```

Specifying the `+version` option (entering `f90 +version`) displays compiler version information to standard output without compiling.

All users can access HP's IT Resource Center on the World Wide Web, which permits searching for bug descriptions and available patches. This is available at the following url: <http://itrc.hp.com>

Related Documentation

The following documents are available for your use in understanding the HP Fortran compiler. Please note that this release note supersedes information in these manuals and white papers.

- *f90(1)* man page, which provides a summary reference to the compile-line options
- *The Libm Library and Floating-Point Arithmetic for HP-UX on Itanium* located at http://ellweb.cup.hp.com/mathlibs/external/document/fp_whitepaper.pdf

NOTE This release note is also available for viewing and printing in the following location and file formats:
</opt/fortran90/newconfig/RelNotes/Fortran90.2.8.html> | pdf | ps | txt.

Current Fixes

This section describes problems that have been fixed in this version of HP Fortran v2.8:

PA-RISC

The following resolutions are for PA-RISC systems:

PHSS_29512

- Wrong answers were given at high opt levels with wide mode.
- There was a problem running `bbcombine` unless software pipelining.
- There was an occasional compiler assert in FE.
- The compiler occasionally aborted with `CHARACTER`, `POINTER`, and `DIMENSION` initialized fields.

PHSS_29796

- There was an Openmp private abort problem.
- There was an occasional assert in the bridge.
- At +O3, after inlining all uses of a module private routine, the compiler terminated with signal 11.
- There was an occasional compiler abort at +O3.
- Wrong answers were given at +O2 and above for `MATMUL (TRANPOSE (S) , S)`.
- There was a `dataval` name issue.
- There was a `rsqrt` performance issue.

New DEV fixes

- When an array index contained a variable with type qualifier, compiling at +O3 +DD64 would give an assert.
- Two array declarations such as `integer ar(100)` and `integer br(100)` produced two unique array declarations, when only one was needed.
- Nested subroutines produced erroneous addresses to break on when the compiler resumed generating code for the body of the containing routine.

Itanium®-based platforms

The following resolutions are for Itanium®-based systems:

PHSS_29663

- Out-of-bounds access to an array stored in a common block yielded wrong answers at high optimization levels.
- Wrong answers were occasionally received at +O3.
- There was an issue with POINTER arrays.
- Occasionally, `declare_F90_F_STOP` did not return.
- SPECcpu2000 occasionally failed to collect the `flow.data` file.
- There was a performance loss of implementation of a matrix multiplication.
- Occasionally, there was a compiler internal error with `OMP PARALLEL PRIVATE` equivalence variables.
- A module with a declared zero size would abort with +DD64.
- The compiler occasionally asserted in FE.
- There was a compiler abort with `CHARACTER`, `POINTER`, and `DIMENSION` initialized fields.

PHSS_29992

- There was an Openmp private abort problem.
- At +O3, after inlining all uses of a module private routine, the compiler terminated with signal 11.
- There was an occasional compiler abort at +O3.
- Wrong answers were given at +O2 and above for `MATMUL (TRANSPOSE (S) , S)`.
- When using +O2, the compiler generated code which resulted in a memory core dump.
- There was a `dataval` name issue.
- There was a `rsqrt` performance issue.
- The allocated memory size of common block differed by the way of declaration.
- There was an occasional middle end assertion.

Fixes not present in prior patch releases

- There was an issue with math inlining suppression for outlined parallel/workshare regions.
- There was an HLO assertion failure at +O3.
- When an array index contained a variable with type qualifier, compiling at +O3 +DD64 gave an assert.
- Two array declarations such as `integer ar(100)` and `integer br(100)` produced two unique array declarations, when only one was needed.
- Nested subroutines produced erroneous addresses to break on when the compiler resumed generating code for the body of the containing routine.

Previous Fixes

The following subsections, “Fortran90 on HP-UX/PA-RISC” and “Fortran90 on Itanium-based platforms”, describe problems that have been fixed in previous versions of HP Fortran for HP-UX:

Fortran90 on HP-UX/PA-RISC

The following list describes problems that have been fixed and included in previous versions of HP Fortran on HP-UX/PA-RISC.

PHSS_28996 (PA-RISC)

- Wrong answers were received at +O3.
- Subroutines ran much slower when compiled at +O3 than at +O2 for `POINTER` and `ALLOCATABLE` arrays.
- `Goto` in OpenMP occasionally caused the compiler to fail.
- There was an exponential allocation problem.
- Occasionally, “private initialized used record” caused an assert.
- Wrong answers were received in last (inout) arg to `date_and_time` intrinsic at +O2 and +O3.

Fixes not present in prior patch releases (version 2.7)

- There was a 10%+ performance improvement if transpose folded into `matmul`.
- Inquire on an open file by name incorrectly padded out strings longer than the file name, resulting in garbage character in the string.
- There was poor scalability for `auto` in parallel.
- When compiling a file not in the current directory with `+cpp_keep`, the filename was given in error and warning messages were incorrect.
- The compiler aborted on large real const (overflows).
- Misaligned store caused `segv`.
- The address of pruned block taken in block was not being pruned.
- The `DEFINE FILE` statement aborted the bridge.

- There was a front end abort on assumed size array.
- There was an unsupported intrinsic SECNDS(real*8).
- There was a TCG assert in bld_iltype_array.c.
- The compiler aborted on a TSRUCTCON.
- There was a failure to deallocate memory.
- There was a need to handle TDECLARE allocatable DVAR as a pointer.
- There was an unraveled type from ARRAYTYPE.
- There was a wrong answer due to optimizing reductions even when +Ofitacc was specified.
- A Caliper error sometimes resulted with errant unwind information; should be backed out.
- There was a blockstack overflow.
- DFIELD was not found.
- There was a zext assert.
- '\$OPTIMIZE level X off' should default to command line opt level setting.
- Occasionally, Wcount1 failed.
- +objdebug was missing line numbers.
- There was a tcg assert on ICUP.

PHSS_28913

- There was occasional poor performance of POINTER formal arguments.
- FORM=UNFORMATTED did not correctly handle big/little-endian conversion.
- There was occasional poor performance with +Oopenmp.
- There was an assert on ICUP.

PHSS_28696

- Wrong answers occurred in wide mode large routines.
- +O2 caused integer arithmetic loop logic errors.
- Compilation occasionally aborted with *Compiler Internal Error.
- There were incorrect Caliper line numbers for +O2 -g.

Previous Fixes

- There were incorrect offsets for vax structures.
- The compiler rejected *size (dimension) declarations.

PHSS_28559

- Performance was degraded due to the absence of no parms overlap assertion.
- +O2 occasionally caused an error.
- There was a memory leak in the application program.
- An error—*bad stackf90—occurred with +Oopenmp.

PHSS_28486

- Multiplication by -1 and FNEG gave different answers for real*16.
- The compiler occasionally generated extra local variables as evidenced by looking at the stack in the debugger.
- The compiler occasionally aborted with -g on a structure with unions.
- There was incorrect runtime behavior when using +i8.
- When compiling some f90 codes with +O3 +Oopenmp, the compiler asserted after inlining a routine that was passed a record containing a dope vector.
- The address of a member of a nested union was incorrect.
- An error occurred with +Ofaster +O3.

PHSS_27974

- There was inferior runtime performance for POINTER and TARGET arrays.
- The compiler aborted when processing extraneous enddo statements.
- Wrong answers were given with common blocks and thread local variables.

PHSS_27735

- The compiler aborted when an integer was used as a case selector for a logical expression.
- Debug information for array pointer upperbound was incorrect.
- The compiler asserted after inlining a routine that was passed a record containing a dope vector.
- Missed exceptions occurred when running exception handling codes compiled at +O3.

- The compiler asserted when function ENTRYs contained assorted types in wide mode.
- The compiler moved code around OMP lock calls.
- When compiling a file not in the current directory with `+cpp_keep`, the filename given in error and warning messages was occasionally incorrect.
- Default `cpp` options needed to be differentiated between PA and IA.
- When using `+asm/-S` in combination with `+P/+Oprofile`, `+asm/-S` was ignored.
- The compiler occasionally aborted with OpenMP and `PREFETCH` directives.
- The compiler aborted when `+i8` was used with values greater than 2^{**31} .
- The compiler issued an error message when the first argument to an elemental procedure was not an array.
- Extensive use of automatic arrays in a subroutine with a very high call count resulted in performance degradation due to using `malloc` for automatic array allocation.
- There was a segmentation fault compiling incomplete program.
- There was a segmentation fault compiling `f90` program with function returning pointer to array of derived type containing a private typed object.
- The compiler aborted with a `+DO` switch.
- The `c$dir_sync_routine` directive was ignored with warning if `+Oopenmp` was used.
- The compiler aborted when compiling the declaration of a deferred-shape array parameter.
- A `BOZ` constant caused a compiler abort in case statement.
- The compiler occasionally aborted when a parameter was equivalenced to a local variable.
- There was a memory leak when a `save` statement was used with an automatically allocated array.
- There was a compile time error with loop unrolling.
- Wrong answers occurred when passing complex pointers at `+O2` optimization.
- The compiler occasionally aborted when trying to initialize a pointer field of a derived type.

PHSS_28059 (10.20)

- Occasionally, the semantic analyzer was not checking for a derived type prior to accessing the underlying type, thus causing a compiler error.

PHSS_27362 (10.20) and PHSS_27363 (11.x)

- The compiler aborted with the PREFETCH directive inside an OMP DO block.
- The compiler aborted when +i8 was used with values greater than 2**31.
- The compiler issued an internal error message when the first argument to an elemental procedure was not an array.

PHSS_27071 (10.20) and PHSS_27072 (11.x)

- Inlined routines containing static (or Fortran SAVE) variables were giving wrong answers when compiled with +Oparallel or +Oopenmp.
- The compiler aborted at +O3 in replace_interval.
- Wrong answers were received from code within OMP locks.
- Program aborted during runtime in \$\$divoI.
- The compiler aborted during compiling rshift.
- Segmentation fault when executing subroutines with a large number of arguments.
- The compiler issued Error 215, Dummy argument or a private procedure cannot have a type that is private.
- Performance problem caused by extra prefetches when using default cache line size.
- Poor performance with multiple memory moves on some allocatable array types.
- Subscript range error not detected in I/O implied DO.
- The compiled program aborted when named parameters were used in some calls.

PHSS_26862 (10.20) and PHSS_26863 (11.x)

- The compiler did not allow character function results to be used as unit numbers.
- The compiler was generating a bogus error message.
- The compiler failed to setup register r29 correctly.
- The compiler did not call routines to print context during assert.
- The compiler was performing an unnecessary error check.
- The compiler was performing an illegal check.
- The compiler was not ignoring C\$Par when +Oopenmp is used.

PHSS_26731 (10.20) and PHSS_26732 (11.x)

- Improved alias analysis for allocatable and automatic arrays.
- Fixed symbolic handling of min/max expressions to avoid exponential expression growth.
- Suppressed check when the array is null.
- Fixed branch->skip conversion in wide mode.
- A better check was provided to autopromote entity-type.
- Increased the nametable size from 128 to 1024.
- Prevent temporary copy of descriptor base address.
- MIN1 and MAX1 calls are now converted internally to MIN0 and MAX0 when integer arguments are specified.
- The compiler did not look for the alias name for function and subroutine interfaces.
- The compiler aborts with redefinition of macro type.
- Updated the compiler to accept label formats.
- The compiler maintained context beyond the statement function context.
- Emit subtype alignment of 16 bits for item.

PHSS_26376 (10.20) and PHSS_26377 (11.x)

- The optimizer was hoisting a load without hoisting the associated store instruction, creating an uninitialized use of a reference.
- The compiler was incorrectly calculating map region offsets.
- The compiler was incorrectly optimizing away a variable that was still in use.
- The compiler did not correctly output the SLT entry before the exit.
- The compiler needed to extend array(1) to the end of the commonblock for alias purposes.
- The compiler did not default to the correct type under the +i2 flag for BOZ constants.
- The compiler should have issued a meaningful error diagnostic message.
- The compiler's type checking was too strict and has been relaxed for IAND calls.
- The compiler's type checking was too strict and has been relaxed for intrinsic calls.
- The mechanism for handling the Open MP ORDERED directive needed to be enhanced.
- The compiler has been changed to handle OPTIMIZE directives.

Previous Fixes

- The compiler has been changed to handle SAVE_LOCALS directives.
- Fixed symbolic handling of min/max expressions to avoid exponential expression growth.
- The compiler did not issue a diagnostic message for undeclared function reference in a call.
- The compiler did not allow iint to translate to int2 and int4 as appropriate.
- The compiler did not pass int4 dimension to intrinsic.
- The compiler incorrectly generated subscript checks on a zero sized array.
- The compiler incorrectly determined that a variable equivalenced within a common block could be type-size promoted to match the size of address.
- The namespace table was not large enough to handle the source file.

PHSS_26066 (10.20) and PHSS_26067 (11.x)

- The compiler needed to check for valid frame size before running with bad input.
- The compiler assumed loop indices to be integer*4 regardless of the compilation mode, causing unnecessary conversions during runtime.
- An invalid but common idiom of equivalencing length 1 array to the beginning of a common block caused over optimization.
- The compiler was incorrectly optimizing addresses to the LOC function.
- The compiler did not correctly handle a boundary condition when writing a symbol across heaps.
- An error in optimization caused incorrect code generation.
- The compiler was not correctly generating line number references.
- The present intrinsic returned wrong results.
- The compiler was not checking to see if common blocks were indirect within functions and subroutines.
- The compiler generated names incorrectly by using the kind precision instead of the type precision.
- The parser did not allow f77 notation for 2 byte and 4 byte integer constants.
- Error handing getarg and getarg1.
- The compiler generated an error on a valid construct due to incorrect error checking.
- An error in bounds checking prevented valid values as parameters to char.

- Needed to encode the fact that fstream on typeint2 returns int4.
- The compiler had a problem converting Holleriths to integers when using a bit or logical intrinsic.
- The compiler turned on +Olibcalls even though the user specified +Onolibcalls.

PHSS_25858 (10.20) and PHSS_25859 (11.x)

- The compiler was stuck in an infinite recursive call loop when compiling an incomplete type.
- The compiler needed to recognize and correctly handle the \$SHORT directive.
- There was a bug coding the determination of the index intrinsic parameter.
- The +i8 switch was not compatible with some of the OMP internal intrinsics.
- The compiler incorrectly reported the hash line number instead of the line number.
- The compiler failed to convert some types.
- The compiler did not allow dimension as a function result attribute.

PHSS_25771 (10.20) and PHSS_25772 (11.x)

- The CRAY target object was not being handled properly by intrinsics.
- The function result type attributes were not being checked correctly.
- The compiler recognized a hollerith field as an unnamed variable and issued invalid debug information.
- C\$pragma was not recognized, the compiler has been changed to ignore it.
- The compiler was constant folding divides by zero.
- The compiler created temporary variables but failed to privatize them in the parallel region.
- A USE statement of a module which exported an operator definition within a nested scope caused an abort due to failure to clear the pushed operator count.

PHSS_25616 (10.20) and PHSS_25617 (11.x)

- Fixed problem with line number handling.
- Fixed problem with type conversion on array constants.
- Fixed problem with graceful handling of error situation.

Previous Fixes

- Fixed problem with more than one thread saving the result.
- The compiler stored subspace lengths as integers; changed to use unsigned integers.
- Changed the compiler to suppress errors for c\$doacross, c\$\$, *\$\$, and !\$\$.
- Multiple use statements caused a name collision.
- Fixed the compiler to allow C\$\$\$ to be a comment.
- Fixed problem with #line directives throwing off the line counting the original source file.
- Fixed problem with missing label from replicated nodes resulting from parallelization.
- The data limit was too small; increased to 400 megabytes

PHSS_25520 (10.20) and PHSS_25521 (11.x)

- Fixed uninitialized register problem causing intermittent wrong answers at high optimization levels.
- Fixed problem with code to determine where the character length is stored is aborting.
- Fixed problem of over-writing memory due to overflow of 16 bit field describing debugger position.
- Fixed handling of the 'type' form of the print statement so that the label is recognized.

PHSS_25413 (10.20) and PHSS_25414 (11.x)

- The compiler did not correctly infer data type for BOZ (typeless) constants from context.
- Fixed bug in handling of the data statement.
- Fixed the internal compiler assertion.
- Fixed the bug in handling the +Ofaster compile option.

PHSS_25175 (10.20) and PHSS_25176 (11.x)

- Fixed a problem with strength reduction of 64 bit multiplication by unsigned constants.
- Removed an unused warning message "invalid arc calculation".
- Added support for multiple arguments to the +U option.
- To be consistent with f77, updated the compiler to accept 'type' as a synonym for 'print' when not within a declaration.
- Fixed the compiler to resolve the omp_memcpy reference.
- Fixed the OMP intrinsic assertion problem resulting from parameter mismatch.

- Enhanced the compiler to allow up to 300 continuation lines.
- Error in algorithm was causing the compiler to use workload/2 to guide scheduling. Changed this to use workload/numthreads.
- Fixed the internal compiler error generated on a pack intrinsic.
- Fixed the internal compiler error resulting in TCG in_descriptor.c assertion.
- Fixed the compiler internal error.
- Placement of \$HP\$SHARED_COMMON directive contrary to documentation was permitted in F77. Upgraded the f90 compiler to allow the f77 placement.
- Fixed problem with intrinsic handling of dope vector array, resulting in bus error.
- Fixed problem so that OMP directives are not lost.
- Fixed bug in compiler for implicit open of file with unit number greater than 99.

Fixes not present in prior patch releases

- Occasionally, there were problems with type keyword placement.
- There was a TCG in_descriptor.C assert.
- When compiling ftn code, a compiler error resulted.
- An unused warning, “invalid arc calculation,” was removed.
- There was a problem with zero sized array for SELECT CASE (char).
- Private structure was imported.
- Data statement on pointer was using null.

PHSS_ 24771 (10.20) and PHSS_24772 (11.x)

- Occasionally, there was a problem when using +FPD combined with certain optimizations.
- The compiler did not recognize the !\$ALIAS form of the \$ALIAS directive.
- The compiler now returns an error message if an end do statement does not have a matching do statement.
- The compiler was incorrectly calculating the destination address.
- Occasionally, there was a problem with quadword results for entry statements.
- The compiler included the concatenated file plus all of the individual files in the compile and link.
- The compiler would occasionally assert when parallelizing a loop with an inlined routine.

Previous Fixes

- The `cputime` routine did not return the correct value.
- Occasionally, the compiler removed some necessary code.
- The incorrect version of the error message file was built in the compiler.
- There was a problem with the variable step transformation.
- The compiler generated `SAVE` tags for an automatic variable.
- The compiler did not clear the `USEASSOCIATED` bit for use variables which were privatized.
- An `INTERFACE` assignment definition prevented the compiler from resolving an operator.
- The compiler was using the incorrect variable `hashlinenumber`, which did not always track line number.
- When propagating type tags from uplevel imported module to the nested routine, the compiler was not checking to see if the type tags had already been set.
- The compiler inadvertently modified loops not directly attached to `OMP DO` directives.
- The compiler was not correctly typing `I*8` induction variables.
- The compiler was incorrectly using the address of the repetition count instead of the repetition count.

PHSS_23952 (10.20) and PHSS_23953 (11.x)

- The `prefetch` directive occasionally caused an internal compiler error.
- The wrong answer problem with intrinsic function `NINT` was corrected.
- The low level optimization of 64-bit code occasionally caused an internal compiler error.
- When using `+U77`, there was a core dump and memory fault problem.
- Using `-g +z` and `+O2` caused an internal compiler error.
- The compiler was updated to accept `READONLY` as a key word in `OPEN`. This was mapped to `ACTION='READ'`.
- Compile-time performance for large array initialization was improved.
- The `M` and `N` edit descriptors were implemented for `f77` compatibility.
- The compiler aborted in the presence of redundant module uses.
- The compiler was updated to allow the use of the intrinsic “size” function in array declarations.

- Occasionally, the compiler aborted on very large arrays.
- There was a compile-time performance problem with producing debug information at +O1.
- The compiler failed to create threads if the user set CPS_STACK_SIZE too large.
- The compiler now allows an EQUIVALENCE statement to follow variable declaration.
- The compiler was updated to allow % as a comment character (FORTRAN 77 issue).
- The compiler was updated to handle debug symbol indexes above 2**20.
- The compiler aborted on nested routines with ALIAS directives.
- An upgrade of the compiler now results in copy out semantics to optimize for contiguous memory when pointers to arrays are passed as parameters.
- Wrong answers resulted when copying large string constants.
- The compiler now allows "SAVE" to be specified more than once for a variable.
- The Fortran 90 compiler generated different output than the FORTRAN 77 compiler for 2**(-1).
- An internal error resulted when evaluating a parameter statement that used the INT intrinsic.
- The compiler now initializes character variables with length*.
- The compiler calculated incorrect results for -10**30 when compiling using free-format.
- The SAVE statement was fixed so that when no arguments are specified, it does not try to save values not valid for a save list.
- Occasionally, there was a problem when using +O3 +Oall +Oautopar +Oparallel +Ovectorize.

PHSS_23724 (10.20) and PHSS_23725 (11.x)

- The compiler returned an internal error when passing a character substring as a parameter.
- The compiler did not recognize and default to generate code for PA2.0 architecture on L class machines.
- The compiler did not handle ".." in relative source path correctly.
- Occasionally, the shared common directive was not processed correctly.

Previous Fixes

PHSS_23351 (10.20) and PHSS_23352 (11.x)

- The compiler did not allow common blocks to be mapped to system shared memory regions.
- FORTRAN 77 and Fortran 90 processes could not shared memory whose size was not a multiple of 8 bytes.
- The compiler did not unroll a simple loop in a timely manner.
- The Fortran 90 compiler was taking over 100 times longer to compile a series of logical/equivalence statements that the FORTRAN 77 compiler.

PHSS_23243 (10.20) and PHSS_23244 (11.x)

- The compiler did not allow a called “C” subroutine to modify a string literal parameter.

PHSS_23025 (10.20) and PHSS_23026 (11.x)

- The compiler asserted on source with large number of tokens.
- Occasionally, there was an internal compiler error using +Oopenmp.
- Return 0 was not handled properly.
- The compiler now returns an error for the non-supported Complex(16).
- The use of the PRIVATE directive caused an internal compiler abort with bad dictionary reference.

PHSS_22538 (10.20) and PHSS_22539 (11.x)

- Fortran did not inline x^{**r} , where r is a real constant with an integral value.
- The basic block optimizer in the LLO disposed of a store that it incorrectly determined was redundant.
- The optimizer tried to parallelize a loop with multiple exits and aborted. This type of loop cannot be parallelized. A warning message is now generated by the compiler and continues to compiler without parallelizing the loop.
- The compiler was producing an unexpected type of initializer for an array of derived type when the initial value was an array constructor composed of structure constructors.
- The compiler was using an incorrect memory area when several reduction variables were needed within a loop.
- The compiler now generates an error message when unable to handle a disallowed variable.

- The link process caused 2.0 libraries to get pulled in. 1.1 libraries are now pulled in.
- The compiler now supports `+multi_open` to allow a file to be used in multiple Fortran OPEN statements.

PHSS_22464 (10.20) and PHSS_22465 (11.x)

- Internal compiler tables were increased in size to resolve Compiler Internal Errors referencing `f90numtab` overflow with large data initialization.
- The compiler failed to handle an extra set of parentheses in a character variable as an intrinsic argument.
- IXOR of `logical*1` was not supported.
- A new switch, `+io77`, was added to support E and G format treatment of leading zeros to match FORTRAN 77 compiler output for easier comparison of prior results.
- `+fastallocatable` caused errors with allocatable arrays that were SAVED.
- There was a Compiler Internal Error when a module defined a COMMON block and USED another module that also defined the same COMMON block.

PHSS_22290 (10.20) and PHSS_22291 (11.x)

- The compiler failed to correctly handle a constant argument to `sizeof`.
- EQUIVALENCE statements with shared common were not handled correctly by the compiler.

PHSS_22112 (10.20) and PHSS_22113 (11.x)

- Some OpenMP directives caused compiler internal errors when used with Modules.
- When multiple load options occurred in a single compile line and the later options were shorter than the earlier options, incorrect behavior occurred.
- The zero-based `getarg` solution provided by PHSS_20578 caused incompatibilities for some customers using shared library calls to `getarg`.
- FSTREAM intrinsic only returned the lower 32 bits of FILE *fp pointer, causing problems for applications using wide mode (`+DA2.0W`).
- Hollerith literals that extended beyond a single line behaved differently in FORTRAN 77 `+es` than with Fortran with `+extend_source`.
- Request for closer correspondence of I/O output between FORTRAN 77 and Fortran 90.

PHSS_21787 (10.20) and PHSS_21788 (11.x)

- The use of OpenMP runtime routines not yet available (such as `omp_get_thread_num`, `omp_set_lock`, `omp_unset_lock`, `omp_test_lock`) caused an abnormal exit from the compiler instead of returning an appropriate error message.
- There were instances of a segmentation fault in the Fortran front end after an invalid alternate return was detected.
- The use of `+fastallocatable` building module occasionally returned an error.
- Previously, there was a problem with SPEC 191.fma3d.
- The compiler aborted when a `PARAMETER` value was placed in a `CHARACTER` declaration.
- Assigning 65535 to an `integer*2` generated an error message.
- Parallel reduction overflows were not handled correctly.
- There was a difference in literal printing between `f77 +es` and `f90 +extend_source`.
- Issues with Union overlapping other variables were resolved.
- Assigned format labels in wide mode were not being handled correctly.
- `LOGICAL FUNCTION G*1 ()` syntax was not accepted by `f90`, but was accepted by `f77`.
- The OpenMP directive error handling was inadequate.

PHSS_21485 (10.20) and PHSS_21486 (11.x)

- Integer exponentiation of negative numbers by negative numbers resulted in incorrect results.
- Some variables beginning with `Z` in data statements were not being handled properly.
- Logical statement functions containing floating points returned incorrect results.
- Occasionally, there was a backend assert while compiling `+O3 +Oparallel` for a loop that had a multiple of 2 loop stride.
- Alternate return arguments in an external subroutine call that was part of an `IF` statement caused an internal compiler error.
- `REAL*4` constants that exceeded the range of `REAL*4` variables caused a compiler time error to be generated.
- There was a need to support the OpenMP model of `threadprivate`.

Fortran90 on Itanium®-based platforms

The following list describes problems that were fixed in previous versions of HP Fortran for Itanium®-based platforms.

Fixes not present in prior patch releases

- Occasionally, compilation was slower at +O3 than at +O2, apparently due to the +Oloop_unroll_jam default set to “on.”
- An assertion occasionally occurred if an indirect call omitted the alternate return entry on the first call.
- WMATMUL at +O0 added special case for a scalar assign guard that looks like an array assign.
- Occasionally, wdb did not find local variables with “info locals” because variable scope was declared to be a module rather than a routine.
- Computed goto label references were not relocated for the cloned IF region associated with an OMP PARALLEL IF.

PHSS_28914

- FORM=UNFORMATTED did not handle big/little-endian conversion.
- Occasionally, there was an assert on ICUP.
- wdb did not find local variables with ‘info locals’.

PHSS_28697

- Occasionally, the compiler would abort due to emitting DSTRUCTON.
- There were incorrect Caliper line numbers for +O2 -g.
- There were incorrect offsets for vax structures.
- Occasionally, the compiler rejected *size(dimension) declarations.

PHSS_28560

- Performance was degraded (by 2-4x) when the no parms overlap assertion was no longer present.
- There was a memory leak in the application program.
- An error—*bad stackf90—occurred with +Oopenmp.

PHSS_28487

- The compiler appeared to be generating extra local variables, as evidenced by looking at the stack in the debugger.
- The compiler aborted, error 8901, with -g on a structure with unions.
- When compiling some f90 codes with +O3 +Openmp, the compiler occasionally asserted after inlining a routine that was passed a record containing a dope vector.
- There was an internal compiler error at +O2.
- The address of a member of a nested union was incorrect.
- The compiler occasionally aborted when compiling +O2 and +DSitanium2.
- There was an internal compiler error, TCG assertion.

PHSS_27975

- A 4 byte variable forced to 16bit alignment using equivalence caused TCG assert on size versus alignment.
- The compiled program occasionally aborted when compiled with +O2.
- The compiled program sometimes did not work when compiled with +DSitanium2.
- The compiler asserted in f95store.c on private type in boundless array of derived type.
- Wrong answers were occasionally given when code was compiled at +O2.
- The compiler occasionally aborted when processing extraneous enddo statements.

PHSS_27281

- The compiler occasionally aborted when an integer was used as a case selector for a logical expression.
- Backend optimizer warning messages were using the linker name for routines (with the trailing underscore), not the name as it appeared in the original source.
- The compiler occasionally asserted after inlining a routine that was passed a record containing a dope vector.
- There were missed exceptions when running exception handling codes compiled at +O3.
- The compiler sometimes generated inefficient code at +O2 due to missing floating point normalization code.
- The compiler moved code around OMP lock calls.

- There sometimes was a segmentation violation during cloning of a vararg routine.
- When compiling a file not in the current directory with `+cpp_keep`, the filename given in error and warning messages may be incorrect.
- `+Oparallel` disables speculative loads in parallel regions.
- Default `cpp` options were identical between PA and IA; they should be different.
- When using `+asm/-S` in combination with `+P/+Oprofile`, `+asm/-S` were occasionally ignored.
- There sometimes were incorrect answers with optimizations greater than `+O1`.
- There was an internal failure in low level optimizer, `cm_misc.C`.
- The compiler occasionally aborted with OpenMP and the `PREFETCH` directive.
- Fortran had options to select shared `libF90` and `libU77` libraries, but no corresponding option for `libIO77`.
- The compiler occasionally aborted when `+i8` was used with values greater than $2^{**}31$.
- The compiler sometimes issued an internal error message when the first argument to an elemental procedure was not an array.
- The compiler occasionally aborted with an error in the `Arith.C` routine of the low level optimizer.
- The compiler sometimes generated inefficient code at `+O3 +Oparallel`.
- There were wrong answers from generated programs at optimizations greater than `+O1` when using `-K`.
- Wrong answers were given from generated programs at optimization levels greater than `+O2` with or without `+Odataprefetch`.
- Extensive use of automatic arrays in a subroutine with a very high call count resulted in poor performance due to using `malloc` for automatic array allocation.
- There was a segmentation fault compiling incomplete program.
- There was a segmentation fault compiling `f90` program with functions returning pointer to array of derived type containing a private typed object.
- The compiler aborted with sanity test failure in `DataPrefetch`.
- The compiler aborted with `+i8` in `TMDATE`.
- The compiler encountered an internal error in `ModSched.C`.
- Compiling with `+Oparamoverlap` resulted in segmentation fault.

Previous Fixes

- Compiling with +Oparamoverlap resulted in low level optimizer failure.
- The loop unroll factor was not being honored on loops which are inlined or cloned.
- Compiling with +Ofenvaccess resulted in compiler abort.
- The directive c\$dir sync_routine was ignored with a warning if +Oopenmp was used. It should be supported regardless of optimization levels.
- The compiler did not catch illegal parameters on a deferred shape array.
- Wrong answers were given at +O2 optimization.
- BOZ constant caused compiler aborts in case statement.
- There was an internal failure in low-level optimizer, file Domin.C.
- There was an internal failure in low-level optimizer, file PredAnaly.C.
- The compiler occasionally aborted when compiling with both the +Onoinline and +Oprofile options.
- The compiler aborted when a parameter was equivalenced to a local variable.
- There was an internal failure in low-level optimizer, file SchedDriver.C.
- Wrong answers with +O2 +DSitanium2.
- There was a memory leak when a save statement was used with an automatically allocated array.
- Wrong answers were generated when passing complex pointers at +O2 optimization.
- There was a performance problem due to loss of loop unrolling.
- The compiler occasionally aborted when trying to initialize a pointer field of a derived type

PHSS_27073

- Inlined routines containing static (or Fortran SAVE) variables were giving wrong answers when compiled with +Oparallel or +Oopenmp.
- Wrong answers were given from the code within OpenMP locks.
- The compiler aborted during compilation of rshift.
- The compiler issued an error that a dummy argument of a private procedure could not have a type that was private.
- There was occasional poor performance with multiple memory moves on some allocatable array types.

- The compiler occasionally aborted when named parameters were used in some calls.

PHSS_26864

- The SIGXCPU signal did not abort programs that had the UNIX95 environment variable set when the programs were linked.
- The compiler generated an error message for `read(f(),*)` when `f` was a character function.
- The compiler generated an error message for a labelled ENDDO statement.
- Traceback information did not always output when the compiler aborted.
- The compiler generated an error when a statement function was used for a character declared with a parameter.
- The compiler processed C\$PAR directives even when OpenMP was used.

PHSS_26733

- Occasionally, there was inferior performance for automatic and allocatable arrays.
- Compiling at +O3 sometimes resulted in the compiler running out of memory in codes that accumulate a result from a large number of min or max expressions.
- f90 option -C - reported an error with zero-sized array.
- Implemented +A alignment option for backward f77 compatibility.
- There were occasional wrong answers at +O2 +DA2.0W.
- The compiler sometimes aborted with internal error 1000.
- The compiler sometimes aborted with name table overflow.
- The compiler occasionally had wrong answers at +O3 with automatic arrays.
- The compiler issued an error message when MIN1 and MAX1 were called with integer arguments.
- The \$HP\$ALIAS directive on interfaces was not always being honored.
- The compiler was looking up constant operands in the dictionary table.
- The compiler aborted on use of a format label with OpenMP; legal code was rejected.
- The compiler gave wrong answers in 64 bit mode when using statement functions.
- The compiler aborted for +A compilation of a 2 byte aligned common item.

PHSS_26378

- Occasionally there were extremely long (infinite) compiles on case statements.
- When using allocatable arrays embedded in a f90 record type, the program would bus error or seg fault during the program execution.
- The compiler aborted when -g was specified with union map regions containing only one component.
- The program received wrong answers at +O2.
- GDB could not stop at the end of a program due to missing SLT entry.
- The executable received wrong answers with parmsoverlap at optimization level +O3.
- Request for O'400' and Z'400'to be treated as 2 bytes for intrinsic calls.
- The compiler aborted when a PARAMETER statement redefined a COMMON definition.
- The compiler rejected IAND (I4, i2const) with +i2.
- IAND and IOR did not accept a mixture of INTEGER*2 and INTEGER*4 arguments.
- The compiler mishandled the ORDERED clause within nested parallel directives.
- The compiler did not honor f77 style OPTIMIZE directives.
- The compiler did not honor f77 style SAVE_LOCALS directives.
- Compiling at +O3 resulted in the compiler running out of memory in codes that accumulate a result from a large number of min or max expressions.
- The compiler aborted when an undeclared function reference was used in a call.
- Intrinsic iint was not allowed with type int.
- The compiler aborted with +i8 on SIZE(array, dim) intrinsic.
- The compiler generated runtime checks causing the program to abort when -C was used with zero-sized arrays.
- The compiler aborted on NASTRAN code.
- The compiler aborted on very large source files.

PHSS_26068

- The compiler aborted with an internal error with large real*8 arrays.
- Inefficient code was generated in 64-bit mode for loop indices.
- The compiler issued internal error messages.

- A wrong answer was received at optimization level +O2 due to use of EQUIVALENCE.
- A wrong answer was received at optimization level +O2 when using the LOC intrinsic.
- The compiler was slow to compile case selectors with a range.
- The compiler issued a bogus error.
- The compiler generated wrong answers in wide mode at optimization level +O3 when using integer*8 induction variables.
- The debugger was not setting breakpoints in functions included via header files.
- The present intrinsic returned the wrong result when an optional parameter was not passed by the calling program.
- Incorrect code was generated when an indirect common block was used in a subroutine.
- The compiler generated incorrect names for complex types for use by the debugger.
- There was an f77 compatibility issue regarding nnnnl and nnnJ notation.
- The compiler generated an internal error on call to getarg when using +getarg1.
- The compiler occasionally aborted with an assertion error.
- The compiler generated an error when values greater than 127 were passed to the char intrinsic in a PARAMETER statement.
- The compiler generated low-level optimizer internal error messages.
- The compiler aborted with a segmentation violation when a function with alternate returns was called.
- The compiler aborted on fstream with integer*2 result assignment.
- There was an error compiling hollerith constants when used with the IAND intrinsic.
- The compiler ignored +Onolibcalls at optimization level +O2.

PHSS_25860

- The compiler aborted when parallelizing OpenMP programs with parallel sections.
- The compiler aborted in a segmentation fault when compiling an incomplete type at +O3.
- The compiler did not recognize the \$SHORT directive, which is needed for f77 backward compatibility.
- The compiler aborted using +i8 when compiling the index intrinsic.
- The compiler aborted in wide mode with the +i8 switch when compiling OMP intrinsics.

Previous Fixes

- The compiler reported the wrong line number in error messages when the -I include path is used.
- The compiler gave incorrect results for integer*1.
- The compiler aborted compiling set_num_threads call using +autodbl in 64-bit mode.
- The compiler issued an error 300 message when compiling a generic function.
- The compiler issued an error message for a character array argument whose length was different than a specific formal parameter character array argument.

PHSS_25724

- The debugger did not stop at the correct line number.
- The compiler aborted with segmentation fault after correctly reporting errors.
- Random wrong results sometimes occurred in Open MP code.
- The compiler did not correctly handle BOZ constants in intrinsic function calls.
- Code produced at optimization level +O2 aborted with a bus error.
- There was a compiler error when compiling a recursive function; abnormal exit from bridge.
- The compiler aborted when the source file contained only comments and blank lines
- The compiler asserted with "Backend Error" when PARAMETER was used with RESHAPE.
- The compiler did not accept the incremental linking options +ild and +ildrelink.
- The compiler asserted on the data statement which uses rep and has another overlapping data statement
- The compiler aborted with f90: Signal 6
- Use of the +Ofaster compile option caused a bus error
- There was a problem shifting a 64 bit integer by 64 bits
- The compiler gave an internal error with array assignment.
- Wrong answers were occasionally received for reentrant subroutines
- The use of 300 continuation lines sometimes caused internal compiler error.
- The label on a 'type' statement was lost.
- The compiler aborted when -C was used

- The compiler aborted with a low-level optimizer message
- The compiler was changed to suppress errors for `c$doacross`, `c$$`, `*$$`, and `!$$`.
- The compiler emitted a warning for an OMP parallel directive.
- The compiler aborted when a module was used multiple times within the same source file.
- The compiler issued an error message for `c$$$`.
- The compiler generated a missing label assertion on Open MP code.
- The compiler appeared to limit initialized DATA to 42 megabytes, much smaller than F77.
- The compiler emitted an assert on operator in INTERFACE definition.
- The compiler asserted with `-g` on a program with a hollerith constant.
- The compiler issued an error with `+Openmp` a pragma directive.
- There was a problem with compile-time evaluation of math function calls with constant arguments.
- The compiler issued an error message when defining a generic function using an INTERFACE block when the function specification contains a RESULT clause.
- The compiler aborted during handling of a CRAY pointer object.
- The compiler encountered an internal error in low-level optimizer file RaDf.c.

PHSS_25009

- There was a compiler internal error with the parallel directive.
- The `$alias` directive was not being accepted.
- Character declarations which follow the equivalence declarations were syntactic errors due to the grammar specification.
- There was no debugger support for Cray Pointers.
- Wrong answers were received with assigned GOTO compiled at `+O3`.
- A mixed multiple entry caused an abort.
- There was poor optimization of arrays with descriptors at `+O3`.
- Array declaration rejected when it involved specifying array size based on a previous arrays size declaration.
- Certain uses of zero-sized arrays/character variables caused compiler asserts.
- Poor performance sometimes resulted on array sections as actual arguments.

Previous Fixes

- There was occasionally excessive compile time for data statements that create large initialized arrays.
- +Ostore_ordering was not being honored.
- Exact +cat option placement on command line changed behavior.
- Empty directive nests were not being supported.
- The omp_set_num_threads intrinsic did not behave as expected.
- There were problems with -exec/-dynamic/-minshared interactions.
- There was a +DD64 assigned format compiler abort.
- Interference between c\$dir and c\$omp.+ild/+ildrelink was not recognized correctly.
- There was an application bus error.
- There was an assert on multi-dimensional array constant in record.
- The REPEAT constructor compiler asserted.
- Pack intrinsic / parenthesized expression; compiler assert.
- A compiler internal error occurred when importing private structure.
- Wrong answers were received for some parallel loops.
- The compiler sometimes didn't support enough continuation lines.
- SELECT CASE (chartype) caused a compiler assert.
- TYPE statement in declarative region rejected with syntax error.
- OpenMP integer*8 loops were not shared.
- OpenMP loops were not always trip counted.
- There was an occasional abnormal exit from the compiler.
- The source with the includes files was not fully debugable.
- Wrong answer was received at +DD64 +O2 +i8.
- The compiler aborted with privatized USE.
- There was a compiler error with allocatable arrays and the save statement.
- There was a module not found error messages none: Not linking with the IA64 unwind library.
- OMP Copyin clauses were not correct.
- Undocumented M edit descriptor from F77 needed to be supported in F90.

- Externally declared omp callables were not getting the right linkages.
- There was an assert on an unexpected DLABEL.
- OMP callables required treatment as external rather than intrinsic.
- Reduction on subtract incorrectly subtracted.
- There was a memory corruption on large OMP codes.
- There was an occasional problem with automatic arrays privatized in parallel region.
- Some OpenMP reductions were not recognized as valid.
- The OMP fmt label refs caused error.

PHSS_24486

- SEGV was received while compiling at +O4.

Fixes not present in prior patch releases

- Occasionally, there was an optimization problem with the use of the option +O2. Different results were calculated if the binary was compiled with different optimization levels.
- When compiled in 64-bit mode, the executable received a bus error.
- The debug information for arrays of characters passed by descriptor character `cstring_arr (:, :)*(*)` was incorrect.
- Occasionally, v2.3 did not `autopar` parallelize the indicated loop.
- When using `+Onolimit +Odataprefetch +O3 +Oprocelim +Olibcalls`, `ccom` entered an infinite loop.
- Previously, `+Onoloop_transform` inhibited loop parallelization.
- Fortran range checking displayed line #0 instead of the actual line number.
- `+Onoinline` did not work for call sites that were transformed from indirect calls to direct calls.
- Fortran ignored 17 `loop_parallel` directives in LS-DYNA.
- `+DA1.1 +Odataprefetch` generated an internal code error.
- Inlining dropped the register storage class of certain variables.
- The compiler would occasionally have an abnormal termination when compiling at optimization level +O3 or higher.
- Wrong answers were generated when computing polynomials at `+Oparallel`.

Previous Fixes

- Occasionally, inlined code created a situation where loop recognition failed to recognize a perfect nest, thereby disabling a critical interchange. The HLO loop recognition phase has been improved to handle such cases.
- Previously, `+i8 +O2` caused an infinite loop in the compiler.
- Using `+i8` and `+O2` occasionally resulted in an internal error with the Fortran compiler.
- There were instances when the compiler produced an internal error if you were using `'-g'`.
- The Fortran driver was updated to infer `+DA<model>` machine parameters.
- A new command-line flag— `+sharedlibF90` —resolves potential issues with the F90 driver trying to link with the shared version of `libF90`. See *Command-line flags* elsewhere in this document.
- The `PACK` intrinsic has been corrected to properly handle nonstandard `LOGICAL` values. Previously, a segmentation fault would occur if nonstandard `LOGICAL` values were encountered by this intrinsic.
- In previous versions of the compiler, the `MERGE` intrinsic was unimplemented, resulting in compiler aborts. Instances where this happened included when the compiler encountered complicated expressions involving derived types or character strings. The `MERGE` intrinsic has been corrected so that it properly handles complicated expressions.
- (HP-UX 11.0 only) Fortran has been corrected so that it works in wide mode (64-bit addressing) when a derived-type selector is used on an array.
- The `LBOUND`, `UBOUND`, and `SHAPE` intrinsics asserted when their result array was non-contiguous.
- The Fortran compiler previously declared some variables as `thread_private` when they were not. The Fortran memory classes have been changed to correct this.
- (HP-UX 11.0 only) In certain situations, Fortran would abort at `+O3 +Oparallel`.
- In certain situations, Fortran would abort at `+O3 +Opal`.
- (HP-UX 11.0 only) Using `CXperf` profiling support option `+pal` with optimization enabled caused Fortran to assert.
- Using `+autodbl` occasionally resulted in an assert.
- (HP-UX 11.0 only) The HP Performance Analysis Tool `CXperf` can now profile Fortran applications with empty routines and compiled with `+pa`. This is a Fortran fix which corrects the previous problem of a resulting coredump.
- Wrong answers were generated by Fortran at `+O3`.
- When the code specified loop interchange, the compiler would not honor `c$dir no_loop_transform` in certain situations.

- `-g +objdebug` now works in the same manner as `+objdebugf -g` so that `+objdebug` takes precedence.
- The compiler would occasionally run out of memory at optimization level `+O3`.
- `SELECT CASE` statements with `REAL*4` ranges/values failed when `+autodbl4` used.
- Explicit double precision constants (`3.0D4`) were corrupted when `+autodbl` was specified.
- The `ADJUSTL` intrinsic returned wrong answers when its argument was a parenthesized character constant.
- `modulo(x,y)` returned incorrect results when $-1 < x/y < 0$.
- The `ES` format descriptor printed zero values incorrectly.
- A never-referenced variable name which was also a common block name caused syntax errors when used in an `EQUIVALENCE` statement.
- Using 4-byte integer `DO` loop indices in wide mode (`+DA2.0W`) incorrectly inhibited a wide range of optimizations and auto-parallelization.
- Certain uses of variable length character variables with the `POINTER` attribute caused intermittent aborts at runtime.
- `NAMELIST` input of character arrays was not compatible with `FORTRAN 77` (non-standard usage of a scalar subscript but multiple values is now accepted).
- `NAMELIST READS` in wide mode (`+DA.0W`) sometimes corrupted memory and caused unpredictable aborts.
- Some character expressions in a `SELECT CASE` statement were evaluated more than once.
- `SYSTEM_CLOCK` sometimes returned an invalid value.
- Short Hollerith initializers (i.e., `1h3`) in `DATA` statements for `REAL` variables caused an internal error.
- Using an `INTENT(IN)` dummy argument in a subsequent initialization expression sometimes caused an internal error.
- Referencing a character variable via a substring and later as if it were a function caused an internal error. Now a syntax error is issued.
- Routines which indirectly used the same module many times caused the compilers symbol table to become corrupted. The compiler now detects and ignores the indirect redundant `USE` statements (it has always ignored directly visible redundant `USE` statements).
- Several internal errors caused by the `+DA2.0` flag have been fixed.

Previous Fixes

- All of the compiler directive code has been reimplemented. Many observed and undiscovered bugs with memory class and loop-based directives have been fixed.
- An unsubscripted integer array used as a FORMAT is now assumed to be a character string format, and not an ASSIGN statement format.
- Zero-length character type array constructors caused an internal error.
- The `ftnxx` environment variables were ignored in wide mode.
- When a library (.a file) was listed several times on the command line, all but one of them was (incorrectly) removed.
- Cray-style pointers were only 32 bits in wide mode unless `+autodbl [4]` was used. Now they are always 8 bytes in wide mode.
- In previous versions of Fortran, space deallocation could potentially fail. This occurred when initialized commons were used in the presence of equivalences.
- Additional NULL pointer checks have been added to the Fortran compiler. This fixes compile-time aborts in the optimizer portion of the compiler.

Software Availability in Native Languages

There are no non-English translations for HP Fortran v2.8 for HP-UX 11i Version 2.0.