

HP Fortran v2.7 for HP-UX 11.0 and 11i Release Note



i n v e n t

Manufacturing Part Number: 5187-2722

0603

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.

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.

1 HP Fortran v2.7 for HP-UX 11.0 and 11i Release Note

Announcement

HP Fortran v2.7 for HP-UX 11.0 and 11i provides a common sourcebase for all HP-UX operating systems. This release of the Fortran 95 compiler corrects defects and provides object-oriented feature optimizations. New features and bundle contents have been added to enhance the performance of the compiler on both PA-RISC 1.1 and 2.0 architectures.

What's in This Version

This version of the HP Fortran distribution package consists of the following components:

- Full Fortran 95 compiler (based on International ANSI/ISO standards) for PA-RISC
- *New directive/option:* optimize openmp [on|off]
+Oautopar (revised description)
- OpenMP v2.0
- Support for the HP WDB Debugger
- Support for math intrinsic inlining
- Standard Fortran library

New directive/option

Following is a new directive and a revised description for an existing option:

`optimize openmp [on|off]`

- Soptimize openmp on
Soptimize 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 +Oopenmp or +Onoopenmp per each file.

`+Oautopar`

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

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* (B3909-90011).

HP WDB Debugger

The HP WDB 2.1 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 11.0 and later. For additional information, please visit <http://www.hp.com/go/wdb>.

Common sourcebase

HP Fortran v2.7 for HP-UX 11.0 and 11i consists of a common sourcebase of HP Fortran for all operating systems.

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 that 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.

Documentation errors

Following are current known documentation errors to be corrected in subsequent releases of the HP Fortran documentation set:

FTN_IO_BUFSIZ

FTN_IO_BUFSIZ is not currently available with HP Fortran. Therefore, the reference to FTN_IO_BUFSIZ in the *HP Fortran Programmer's Guide, 3rd edition (B3909-90009)* on page 86 should be ignored.

FLUSH

The *HP Fortran Programmer's Guide (B3909-90009)* on page 211 incorrectly makes reference to FLUSH in libU77. FLUSH has never existed in libU77.

Compiler limits

Information on the following compiler limits will be added to the *HP Fortran Programmer's Guide*, Chapter 10 at the next release of the manual:

HP Fortran supports format field widths up to 32767. HP F77 allows format field widths up to 2147483639 which is close to 2^{31} . This chapter also discusses incompatibilities.

Known Problems and Workarounds

These are known problems and workarounds for previous versions of 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.
- The wait system calls in the Fortran compilers generate different statuses when an unpreinstrumented Fortran application is run from CXperf. This is a compiler-specific issue, and not a problem with CXperf. If an application runs normally and returns `status=child process terminated normally with a return code of zero`, then a run from within CXperf will return `*stat_loc` or `status=child process terminated with a SIG_TRAP`. This is expected since the CXperf process interface uses a trapping mechanism to probe the application processes.
- 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 64-bit applications, 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 `+DA2.0W`, 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 `+DA2.OW` 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`

- Both the `+pa` and `+pal` options are ignored when the HP Fortran 90 compiler generates position-independent code (PIC). As a result, the following options cause `+pa` and `+pal` to be ignored: `+pic=short`, `+pic=long`, `+z`, and `+Z`.
- 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.
- Program optimization using the `+O4` option 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 `+DA` and `+DS` compile-line options do not accept 1.0 (or any PA-RISC 1.0 architecture models) as arguments because the HP Fortran compiler no longer supports the PA-RISC 1.0 architecture.
- Pointer arithmetic on Cray-style pointers has different semantics on Hewlett-Packard machines than on some Cray machines. HP Fortran 90 implements Cray pointers as BYTE pointers, while some Cray machines use word addressing.
- When `+O3 +Oreport +pal` is used, the Optimization Report contains extra loops with erroneous line numbers.

Workaround: To avoid this problem, use `+Oinfo` in place of `+Oreport`.

- If you installed a patch that modified `sched.models`, you may have received the `+DA1.1` code by default.

Workaround: If this is the case, you will need to specify the `+DA2.0` option on each compile so that the compiler will generate code that will gain the benefits of the PA-RISC 2.0 architecture.

- 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`.

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.

PA-RISC 1.0 Architecture Not Supported

The HP Fortran compiler does not support the PA-RISC 1.0 architecture. This means that the `+DA` and `+DS` compile-line options will not accept the 1.0 argument, and they will not accept the following model numbers (or other PA-RISC 1.0 architecture models) as arguments:

600, 635 645, 800, 808, 815, 822, 825, 832, 834, 835, 840, 842, 845, 850, 852, 855, 860, 865, 870, 890

Refer to the f90(1) man page or to the *HP Fortran Programmer's Guide* for more information on the +DA and +DS options.

NOTE The portable argument to the +DA and +DS options creates object code compatible across all PA-RISC 1.1 and 2.0 workstations and servers. It however does not support the PA-RISC 1.0 architecture.

“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

In this version of the HP Fortran compiler, the NODEPCHK directive is sometimes ignored. The symptom was that programs which use the directive would not parallelize as well as expected.

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

no_loop_transform

The *Parallel Programming Guide for HP-UX Systems* inaccurately states, on page 149, that the `no_loop_transform` directive inhibits parallelization. `no_loop_transform` *does not* inhibit parallelization.

Previously-released Benefits

Previous releases of HP Fortran introduced these improvements:

Command-line flags

- `+multi_open`—enables attaching multiple FORTRAN UNITS to the same file by allowing the file to be opened in multiple Fortran OPEN statements.
- `+io77`—suppresses the generation of the optional leading “0” before the decimal point for real numbers printed with the E and F edit descriptors. FORTRAN 77 suppressed these leading zeros unless the NOSTANDARDIO flag was used. Along with `+gformat77`, the `+io77` flag provides formatted output compatibility with the FORTRAN 77 product.
- `+cat`—concatenates all source files of the same source form together, then compiles the concatenated source all at once. This enables inlining at `+O3` within the concatenated file.
- `-dynamic`—used to generate a dynamically-bound executable.
- `+[no]es`—Similar to `+[no]extend_source` except that character literals and hollerith constants continued across a line boundary are not padded. This option provides compatibility with FORTRAN 77’s `+es` option.
- `+getarg0`
`+getarg1`—these options control the behavior of the `getarg` intrinsic subroutine. `+getarg0` requests the industry standard behavior for `getarg`, where an index value of zero causes the program name to be returned. HP’s FORTRAN 77 `getarg` intrinsic also implements this industry standard convention. `+getarg1` is used to request non-standard behavior, where an index value of one causes the program name to be returned (older releases of HP’s Fortran behaved in this manner).
The default is `+getarg0`.
- `gformat77`—requests the FORTRAN 77 style of formatting a value of zero with the G edit descriptor. Fortran uses an F edit descriptor when the value being written is zero, while FORTRAN 77 uses an E edit descriptor.
- `+what`—prints the what string for the Fortran driver, providing version and patch numbers.
- `+[no]signedzero`—enables signed-zero support. This option forces a floating point value of negative zero that appears as a formatted output list item to be represented in the output record with a leading “-”. This option also changes the behavior of the `SIGN` intrinsic.

Previously-released Benefits

The default is `+signedzero`.

- `+es`—similar to the existing `+extend_source` option, but differs in that it does not pad a character literal which spans a continuation line with blanks out to column 255. Instead, it behaves like FORTRAN 77's `+es` option, and only uses the actual blank characters in the continued line in the source file.
`+es` only affects fixed source form blank padding.
`+noes` disables this behavior (the default is `+noes`, meaning no extended lines, and character literals continued across lines are padded with blanks to column 72, the same as FORTRAN 77's behavior).
- `+fastallocatable`—enables a different representation for allocatable arrays in the object code produced by the compiler. This alternate representation avoids problems in optimizing code containing allocatable array references. Additionally, this alternate representation for allocatable arrays is binary compatible with the old representation
- `+noalign64bitpointers`—disables correct alignment of pointers in derived types when compiling for wide mode (`+DA2.0W`). Earlier releases of HP Fortran improperly aligned such pointers, occasionally leading to runtime aborts. Since this change introduces a potential binary incompatibility, the `+noalign64bitpointers` flag is provided to maintain the old behavior. Users who compile in wide mode (`+DA2.0W`)—and have derived types that contain components with the `POINTER` attribute—should recompile all source files that reference variables of that derived type. Users who have successfully used such derived type variables with older releases, and do not wish to recompile all affected source files, should always specify `+noalign64bitpointers` when compiling affected source files.
- `+Openmp`—allows users to enable the OpenMP Directives. `+Onoopenmp` will disable the OpenMP Directives.
- `+r8`—changes 4-byte real constants, intrinsics, and user variables to 8-byte reals (rather than the 4-byte default).
- `+i8`—changes 4-byte integer and logical constants, intrinsics, and user variables to 8-byte integers (rather than the 4-byte default).
- `+sharedlibF90`—allows users to link the shared version of `libF90` or `libF90_parallel` from `/usr/lib`. This resolves potential issues with the F90 driver trying to link with the shared versions of `libF90`.

In order to invoke `+sharedlibF90`, users must install one of the following three patches on *both* the machine used to link the executable *and* all machines intended to run the generated executable when using `+sharedlibF90`:

— PHSS_25714: 11.00 shared libF90

- PHSS_25716: 11i shared libF90
- *No patch is required for 11i.*

NOTE Failure to install one of these patches will cause users to experience either a linker error if trying to link the executable, or a dynamic loader error if trying to run the executable.

GETARG

The `GETARG` intrinsic subroutine has been modified to be compatible with HP's FORTRAN 77 compiler and other vendors' implementations. This change requires source code changes and recompilation of all source files that call `GETARG`.

The first argument to `GETARG`, the index of the command line argument desired, is now zero based instead of one based. A value of one will now cause `GETARG` to return the first command line argument after the program name, instead of the program name, and a value of zero now returns the program name, instead of blanks.

Users wanting the old behavior can specify the following command line options:

```
+getarg1
```

Gather/scatter prefetch directive

The `C$DIR PREFETCH array` directive was previously added to the HP Fortran v2.3 compiler. This directive obtains prefetches for indirect array references. You are responsible for ensuring that the loop bounds are modified so that no out-of-bound array references are generated. A clean-up loop is typically required to account for references that cannot be prefetched.

The syntax of this directive is:

```
C$DIR PREFETCH list-of-array-references
```

Each array-reference must be syntactically and semantically acceptable as a left-hand side of an `ASSIGNment` statement. Any user array with elements identified in a `PREFETCH` directive does not automatically generate prefetching.

Below is an example of sample use of this directive:

```
C$DIR PREFETCH A(J(I+10))  
  DO I=1,N-10  
    A(J(I))=B(I)+C(I)  
  ENDDO
```

Previously-released Benefits

```
DO I=N-9,N
  A(J(I))=B(I)+C(I)
ENDDO
```

In the above loop, automatic prefetching is still generated for the B, C, and J arrays. For the A array, A(J(I+10)) will be prefetched on the Ith iteration of the loop. Since J(I+10) may not be defined for I+10 greater than N, you will have to manually split off the last ten iterations of the loop.

Fortran 95 features

A new parallel DO statement and construct was added to HP Fortran. The form of this construct is:

```
FORALL forall-triplet-spec-list body-of statements
END FORALL
```

where: forall-triplet-spec-list is a list of the following:

index-name=subscript:subscript[:stride] followed by an optional scalar mask expression.

An example of a FORALL statement is shown below:

```
REAL :: A(10,10), B(10,10) = 1.0
FORALL (I=1: 10, J=1:10, B(I,J) /=0. 0)
A(I,J) = REAL(I+J-2) B(I,J) = A(I,J) +B(I,J)*REAL(I*J)
END FORALL
```

LibU77 routines provide Y2K compliance for HP-UX 11.0

Two libU77 subroutines—datey2k and idatey2k—are now available in the Fortran compiler to handle Year-2000 (Y2K) date-related issues on HP-UX 11.0. The +U77 flag must be issued with both subroutines. Although these are provided for Y2K compliance, it is recommended that the standard date_and_time intrinsic be used instead of these functions.

NOTE The existing date and idate intrinsics do *not* have a Y2K problem. They only return the last two digits of the year. And they will continue to do so after the rollover. The behavior of these intrinsics cannot be modified because existing code may depend on the current behavior.

`datey2k` is designed to replace the `f90` date intrinsic. Its function and arguments are the same as `date`'s, except that the returned string contains a 4-digit year (mm-dd-yyyy) instead of a 2-digit year (mm-dd-yy).

The syntax of the `datey2k` subroutine is as follows:

```
subroutine datey2k(date)
character*11 date
```

`idatey2k` is designed to replace the HP Fortran `idate` intrinsic. `idatey2k` returns the true year in its third argument, as opposed to the `idate` intrinsic, which returns the number of years since 1900 in its third argument.

The syntax of the `idatey2k` subroutine is as follows:

```
subroutine idatey2k(month, date, year)
integer month, day, year
```

The `libU77` routine `idate` has similar functionality to `idatey2k` (it returns the true year), but its arguments are passed differently.

- In code where `date` is referenced, replace `date` with `datey2k`. Also, make sure that `datey2k`'s argument is at least 11 characters in length.
- In code where the `idate` intrinsic (not the `libU77` `idate` routine) is used, replace `idate` with `idatey2k`.

Additional Improvements

- Improved run time performance of Fortran features, particularly allocatable arrays. `+fastallocatable` and `+noalign64bitpointers` command-line flags are discussed below in “Command-line flags.”
- A native, subset OpenMP implementation—This new feature implements nine new directives, including `PARALLEL`, `PARALLEL DO`, `PARALLEL SECTIONS`, `DO`, `SECTION`, `SECTIONS`, `BARRIER`, `CRITICAL`, and `ORDERED`. More detailed information about this new feature is supplied in Appendix B of the *HP-UX Parallel Programming Guide*.
- Improved optimization flag defaults.
- `wdb` V2.0 support.
- Improved compiler support—The compiler now supports the complete Fortran 95 standard.

Previously-released Benefits

- **DA1.1 Data Prefetching** — This feature is beneficial if you want to run a single `+Odataprefetch` executable on both PA-RISC 1.1 and 2.0 architectures. Previously, if you built a `+Odataprefetch` executable on a 2.0 machine, the 1.1 architecture ignored it. The `+DA1.1 +Odataprefetch` flags allow programs to be built to run on both the PA-RISC 1.1 and 2.0 architectures but with the same benefits as 2.0 prefetching on 2.0 architectures. These flags cause DA2.0 prefetches to be inserted into the code in the same place they would be for `+DA2.0 +Odataprefetch`.
- **Sum Reduction Optimization** — These have been optimized to compute several partial sums to more efficiently use PA-RISC 2.0 architecture, resulting in 2x runtime improvements. This optimization is only available at `+O2` or higher optimization levels.
- **V2600 support** — Support for V2600 server cache information has been added (HP-UX 11.0/11i only).
- **New Directive**—The `MIXED_FORMATS` directive, which exists in HP FORTRAN 77, now works in HP Fortran v2.3 and above. This directive turns on or off mixed formats in formatted input/output (I/O) statements. It allows a numeric edit descriptor of a type different from that of the corresponding I/O list item to be read from or written to.

The syntax of the `MIXED_FORMATS` directive is: `HPMIXED_FORMATS [ON | OFF]` where `ON` turns mixed formats on in formatted I/O statements, and `OFF` turns mixed formats off in formatted I/O statements.

- **Build-time improvements** at optimization level `+O3`.

System Compatibility Information and Installation Requirements

This section describes system requirements for the installation and running of HP Fortran v2.7 for HP-UX 11.0 or higher on PA-RISC.

Software requirements

Following are system software requirements for running HP Fortran v2.7 from the HP server hardware:

- These patches are required for HP Fortran v2.7 to correctly run on HP-UX 11.0:
 - PHSS_25718 libcl
 - PHSS_26912 libcps
- These patches are required for HP Fortran v2.7 to correctly run on HP-UX 11i:
 - PHSS_25719 libcl
 - PHSS_26913 libcps
- In order to invoke the `+sharedlibF90` flag (discussed earlier in this document), users must install one of the following three patches on *both* the machine used to link the executable *and* all machines intended to run the executable generated when using `+sharedlibF90`.
 - PHSS_25714 (11.00 shared lib F90)
 - PHSS_25716 (11i shared lib F90)
 - *No patch is required for 11i.*

NOTE	In addition to being required for <code>+sharedlibF90</code> , PHSS_25714 and 25716 address other functionality as well, and are therefore, recommended for all customers.
-------------	--

- The HP Fortran compiler requires approximately 172 megabytes (MB) of disk space. This includes approximately 130 MB for the compiler, with remaining space for other components such as the debuggers.

- On HP V-Class servers, the HP-UX kernel must be configured as described in the *HP V-Class Server HP-UX Configuration Notice*. Using the “V-Class Technical Server” configurable-parameter settings can improve your V-Class server’s performance, especially when running multiple larger applications. The configuration document describes how to use the SAM utility (`/usr/sbin/sam`) to apply the “V-Class Technical Server” tuned parameter set and related issues.

Hardware requirements

HP Fortran v2.7 for HP-UX 11.0 and higher on PA-RISC is supported on HP 9000 Series 700/800 hardware running the HP-UX 11.0, 11i, and HP-UX 11.20 operating systems.

Operating system requirements

HP Fortran v2.7 for HP-UX 11.0 and higher on PA-RISC recommends that HP-UX version 11.0 Extension Pack 9804 (version B.11.00.38) April 1998 or greater be installed for OS platform and version compatibility.

Installation requirements on HP-UX 11.0 and higher for PA-RISC

To install HP Fortran v2.7 for HP-UX 11.0 higher, follow the steps below:

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:

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

Support information for 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:

- B3906BB—HP Fortran Series 700
- B3908BB—HP Fortran Series 800

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 <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
- *HP PA-RISC Compiler Optimization Technology White Paper* (5964-9846E). For a PostScript version of this document, see /opt/langtools/newconfig/white_papers/optimize.ps
- Information in these documents supplements the *HP Fortran Programmer's Reference* and the *HP Fortran Programmer's Guide*, and will be incorporated in those manuals at a future release.

NOTE This release note is also available for viewing and printing in the following location and file formats (*on 11i systems only*):
</opt/fortran90/newconfig/RelNotes/Fortran.2.7> | pdf | ps | txt.

11.0 customers should be referred to this Release Note located at <http://docs.hp.com>.

HP Fortran Patches for this Version

This section describes HP Fortran v2.7 patches for HP-UX 11.0 and higher on PA-RISC.

Patches for HP-UX 11.0

No patches are currently required for 11.0.

Patches for HP-UX 11i

No patches are currently required for 11i.

Fixes

This section describes problems that have been fixed in current and previous versions of HP Fortran for HP-UX:

Current fixes

The following list describes problems that have been fixed and included in this version of HP Fortran v2.7 for HP-UX.

Other

- 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 info; should be backed out.

Fixes

- 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_28059

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

PHSS_27363

- 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_27072

- 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.

Previous version fixes

The following list describes problems that were fixed in previous versions of HP Fortran for HP-UX.

PHSS_26863

- 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_26732

- 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 covered 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_26377

- 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.
- 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_26067

- 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_25859

- 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_25772

- The CRAY target object was not being handled properly by intrinsics.

Fixes

- 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_25617

- Fixed problem with line number handling.
- Fixed problem with type conversion on array constants.
- Fixed problem with graceful handling of error situation.
- 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 count in 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_25521

- 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_25414

- 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_25176

- 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 \$HPSSHARED_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.

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.

Fixes

- 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.

Fixes

- 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.
- 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 Pointees.
- 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.
- 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-diminsional array constant in record.
- The REPEAT constructor compiler asserted.
- Pack intrinsic / parenthised 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 privated 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 sometimes 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.

Other

- 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.

Fixes

- 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.
- 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_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 `+Openmp`.
- 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_23243 (10.20) and PHSS_23244 (11.x)

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

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_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.

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.

Fixes

- 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_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 compile 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`.

Other fixes

- 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`.
- 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.

Fixes

- 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 `+objdebug -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.
- All of the compiler directive code has been reimplemented. Many observed and undiscovered bugs with memory class and loop-based directives have been fixed.

Fixes

- 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 v2.7 for HP-UX 11.0 and higher on PA-RISC.